creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

데이터 증강을 통한 순차 추천 성능 향상 연구

2021 년 2 월

서울대학교 융합과학기술대학원

지능정보융합학과

송 주 영

데이터 증강을 통한
순차 추천 성능 향상 연구

지도 교수 서 봉 원

이 논문을 공학석사 학위논문으로 제출함
2020 년 12 월

서울대학교 융합과학기술대학원
지능정보융합학과
송 주 영

송주영의 공학석사 학위논문을 인준함
2021 년 1 월

위 원 장 _____ 이 교 구 _____ (인)

부위원장 _____ 이 원 종 _____ (인)

위 원 _____ 서 봉 원 _____ (인)

초 록

정보화 시대의 성숙을 바탕으로 발달해 온 추천 시스템은 최근 심층 인공신경망(Deep Neural Network) 구조를 학습에 활용하는 딥러닝(Deep Learning) 기술의 발달과 함께 높은 성능 향상을 이루어내고 있으며, 개선된 성능에 힘입어 그 영향력 및 중요성이 더욱 증가하고 있다. 많은 경우에 심층 인공신경망 구조를 활용한 모델은 전통적인 접근법 대비 높은 성능을 얻을 수 있는 것으로 알려져 있으나, 이를 위해서는 수많은 파라미터들을 추정하기 위한 대규모의 데이터셋이 전제된다. 하지만 기 확보된 데이터셋의 규모가 작은 상황에서 데이터를 추가적으로 확보하는 것은 높은 비용이 소요되며, 특히 실제 사용자의 행동 로그를 다루는 추천 시스템은 타 분야에 비해 추가 데이터 확보가 더욱 어려운 편이다.

딥러닝 기술의 적용이 활발하게 이루어지고 있는 컴퓨터 비전(Computer Vision) 분야의 경우 다양한 데이터 증강(Data Augmentation) 기술을 통해 절대적인 학습 데이터셋의 규모를 크게 늘리는 방식으로 모델의 성능 향상을 도모해오고 있다. 추천 시스템 역시 이러한 데이터 증강 기술을 활용하여 학습 데이터를 추가적으로 확보할 수 있다면 상당한 효용이 있을 것으로 기대되나, 아직까지 추천 시스템, 특히 순차 추천(Sequential recommendation) 분야에서는 데이터 증강 관련 연구가 미흡한 실정이다.

이러한 문제의식을 바탕으로, 본 논문에서는 학습 데이터가 충분하지 않은 상황에서 다양한 데이터 증강 기법을 활용함으로써 순차 추천 시스템의 성능을 향상시킬 수 있음을 보이고자 한다. 이를 위해 인공신경망 기반의 최신 순차 추천 모델을 기반으로 원본 아이템 시퀀스를 직접 변형시키는(direct corruption) 방식의 총 4 가지 데이터 증강 기법 1) 노이즈 추가(Noise Injection), 2) 중복성 추가 (Redundancy Injection), 3) 아이템 마스킹(Masking), 4) 유사 아이템 대체(Synonym Substitution) 을 적용하여 추천 성능의 변화를 확인하였다. 순차 추천 모델의 성능 평가를 위한 벤치마크로 널리 사용되는 무비렌즈(MovieLens-1M), 고왈라(Gowalla), 아마존 비디오게임(Amazon Video Games) 의 3 개 데이터셋에 대해 실험을 진행한 결과, 제안하는 데이터 증강 기술을 적용할 경우 전반적으로 성능 개선이 나타나는 것으로 확인되었다. 특히 데이터셋의 크기가 작을 때에 데이터 증강에 의한 성능 향상의 폭이 큰 것으로 나타나, 순차 추천 시스템이 적용되는 서비스 초창기에 데이터를 충분히 확보하지 못했을 경우 성능 향상을 위해 데이터 증강이 효과적으로 활용될 수 있을 것으로 보인다.

본 연구의 기여 부분은 다음과 같이 정리할 수 있다. 1) 데이터 증강 기술을 활용하여 학습 데이터가 적은 상황에서 순차 추천 성능을 개선할 수 있음을 정량적인 실험을 통해 확인하였다. 2) 데이터 전처리 과정에서 이루어지는 데이터 증강은 기본적인 모델의 아키텍처를 변화시키지 않는다는 점에서 현재 제시되어 있는 다양한 SOTA(State-Of-The-Art) 모델의 성능을 추가적으로 개선시킬 수 있는 가능성을

제시하였다. 3) 다양한 데이터 증강 방식을 포괄적으로 적용해 봄으로써 데이터 증강 방식에 따른 성능 변화의 차별적 양상을 확인하였으며, 데이터 증강이 추천 시스템 디자인에 있어 보편적인 전처리 기법의 하나로 기능할 수 있을 가능성을 검증하였다. 4) 기존의 순차 추천 분야에서 아직 충분히 연구되지 않은 데이터 증강이라는 측면에 주목함으로써 향후 관련 연구들을 발전시키는 과정에서 기초 자료로 활용될 수 있다.

주요어 : 순차 추천, 추천 시스템, 데이터 증강, 데이터 전처리, 딥러닝

학 번 : 2019-21923

목 차

그림 목차

표 목차

수식 목차

그래프 목차

제 1 장 서 론

제 1 절 연구의 배경

하루에도 셀 수 없이 방대한 양의 데이터가 새롭게 만들어지고 있는 현대 정보 사회에서, 추천 시스템은 여러 산업 분야를 넘어 사람들의 일상에 이르기까지 널리 활용되고 있는 핵심 기술이라고 할 수 있다. 오늘날 사람들의 정보 획득 방식은 더 이상 단순히 인터넷, 검색 엔진을 주 무대로 구체적인 질의(query) 를 던지는 능동적인(proactive) '정보 검색' 활동에 국한되지 않는다. 자신이 이용하고 있는 다양한 웹 사이트 또는 모바일 어플리케이션이 개개인에 따라 맞춤형으로 제시하는 개인화된 정보에 적극적으로 반응하는(reactive) '정보 수용' 활동 역시 매우 큰 부분을 차지하고 있으며, 그 기반이 되는 대표적인 기술이 바로 추천 시스템인 것이다.

일반적인 개인화 추천 시스템은 어떤 아이템에 대한 사용자의 선호가 변화하지 않는 고정적(static) 이고 보편적인(general) 성질의 것이라는 가정 아래 사용자와 아이템 간의 관계, 곧 개별 아이템에 대한 사용자의 선호를 모델링한다. 따라서 기존의 많은 추천 시스템들은 사용자가 특정 아이템을 소비(interact) 한 순서나 맥락을 고려하지 않은 일반적인 선호 만을 기준으로 추천 결과를 도출하게 된다. 하지만 동일한 아이템이라 하더라도 사용자가 해당 아이템을

추천 받고 소비하는 시점과 맥락에 따라 사용자가 느끼는 선호의 정도, 곧 만족도는 달라질 가능성이 존재한다. 예를 들어, 평소에는 시끄럽고 빠른 비트의 댄스 음악을 즐겨 듣지 않던 사람이더라도 한밤 중의 교외 드라이브, 친구들과의 흥겨운 술자리와 같은 상황에서 분위기를 북돋을 음악을 선곡하고자 한다면 충분히 댄스 음악을 플레이리스트에 추가할 수 있을 것이다.

이러한 문제 의식을 바탕으로 등장한 순차 추천 시스템 (Sequential recommendation) 은 사용자가 이전에 아이템을 소비한 순서와 맥락을 함께 고려하여 그 다음에 소비할 아이템이 무엇인지 예측하고 추천하는 것을 목적으로 한다. 이는 아이템에 대한 사용자의 선호가 맥락에 따라 달라질 수 있는 동적인(dynamic) 성질의 것으로 간주하고, 해당 아이템 시퀀스의 시계열 패턴 (sequential pattern) 을 파악하고자 한다는 점에서 일반적인 추천 시스템과 차이가 존재한다. 앞서의 음악 선곡 예시를 통해 쉽게 확인할 수 있듯이, 개개인의 취향을 고정적, 보편적인 것으로만 가정한다면 경우에 따라 추천 문제를 지나치게 단순화 할 가능성이 있으며, 이는 결과적으로 추천의 정확도를 떨어뜨려 사용자의 만족도를 저하시키게 될 우려가 있다.

또한 산업적인 측면에서, 순차 추천 시스템은 최근 '정보 수용' 활동의 비중, 영향력이 증가함에 따라 더욱 주목받을 수 밖에 없다. 음악 감상 및 동영상 시청, 뉴스 확인, 쇼핑, 소셜 네트워크 서비스(SNS) 등 다양한 서비스 도메인에서 순차 추천 성능이 향상되면, 곧 사용자가 그 다음에 소비할 만한 아이템을 정확하게 추천하면 사용자를 해당 어플리케이션에 오래도록 머무르게 할 수 있기 때문이다.

전세계 최대 규모의 동영상 공유 플랫폼 ‘유튜브(YouTube)’의 사례가 대표적으로, 이들은 사용자가 특정 동영상을 시청한 뒤 추천 알고리즘을 통해 ‘다음에 이어서 볼 동영상’을 잘 추천함으로써 사용자를 해당 플랫폼의 생태계 안에 묶어두는 ‘락인(lock-in)’ 효과를 누리며 ‘올인원 앱’으로 진화하고 있다.

이처럼 순차 추천은 고전적인 개인화 추천과 비교해 하나의 개인을 보다 복합적인 선호와 취향을 가진 존재로 이해하는 알고리즘으로서, 점차 그 중요성이 커지고 있다고 할 수 있다.

제 2 절 연구의 내용

순차 추천 시스템을 구현하기 위한 고전적인 접근법으로는 마코프 체인(Markov Chain) 기반의 모델이 가장 대표적이며, 최근에는 여러 도메인에서 비약적인 성능 향상을 이끌고 있는 딥러닝(Deep Learning) 기반의 모델들이 다양하게 제안되고 있다.

순차 추천 시스템은 기본적으로 순서가 있는 아이템 시퀀스 (item sequence) 를 입력으로 받아 모델링을 수행한다는 점에서 단어 시퀀스(word sequence), 곧 문장(Sentence) 이 주 입력 단위가 되는 자연어처리(Natural Language Processing, NLP) 분야의 접근법이 활발하게 적용되어 왔다. 특히 해당 분야에서 중점적으로 연구되어 온 순환신경망 (Recurrent Neural Network, RNN) 구조, 어텐션(Attention) 메커니즘과 같은 딥러닝 아키텍처는 순차 추천 시스템에 활용되었을 때에도 뛰어난 성능을 나타내는 것으로 확인되었으며, 이외에도 합성곱신경망(Convolutional Neural Network, CNN), 변분 오토-인코더(Variational Auto-Encoder, VAE) 등을 활용한 많은 딥러닝 기반 모델들이 SOTA(State-of-the-art) 수준의 성능을 기록하고 있다.

이처럼 현재까지 순차 추천 시스템의 구현에 있어 적극적으로 도입되어 온 딥러닝 기반 접근법은 주로 모델 구조(model structure) 개선이라는 측면에 집중되어 있으나, 선행 연구들에 따르면 딥러닝 기반 모델의 성능에 영향을 미치는 요인은 단순히 모델 아키텍처에만 한정되지 않는다.[1] 데이터의 입력(Input), 전처리(Processing) 및 모델

학습(Training) 측면에서의 여러 요인들이 복합적으로 해당 모델의 성능을 결정하지만, 순차 추천 분야에서 이들 측면은 상대적으로 소홀히 다루어져 온 것이 사실이다. 특히, 복잡한 인공신경망(Neural Network) 구조를 갖는 딥러닝 기반 모델이 높은 성능을 얻기 위해서는 학습 과정에서 수많은 매개변수(parameter) 들을 제대로 추정하기 위한 대규모의 데이터셋이 전제된다. 이를 위해 데이터 전처리 측면에서 기존 데이터셋이 지닌 특성을 왜곡하지 않는 수준에서 인위적인 변화를 더해 추가적인 학습 데이터를 확보하는 '데이터 증강(Data Augmentation)' 기술이 성능 개선을 위해 유용하게 활용될 수 있다.

앞서 딥러닝 기술의 적용이 활발하게 이루어져 온 컴퓨터 비전(Computer Vision) 분야의 경우 일찍이 다양한 데이터 증강 기술을 통해 수많은 변형 이미지 데이터를 생성하여 학습 데이터셋의 규모를 늘림으로써 모델의 성능을 향상시키는 접근법이 널리 적용되어 오고 있으며[2,3], 음성 인식(Speech Recognition) 분야에서도 오디오 신호 및 스펙트로그램(spectrogram) 데이터에 대한 다양한 증강 기술이 연구되어 왔다.[4] 자연어처리(NLP) 분야의 경우 일반화된 변형 규칙을 만들어 내기 어렵다는 점에서 상대적으로 데이터 증강에 대한 연구가 더디게 이루어져 왔으나, 최근 간단한 증강 기법의 적용만으로 텍스트

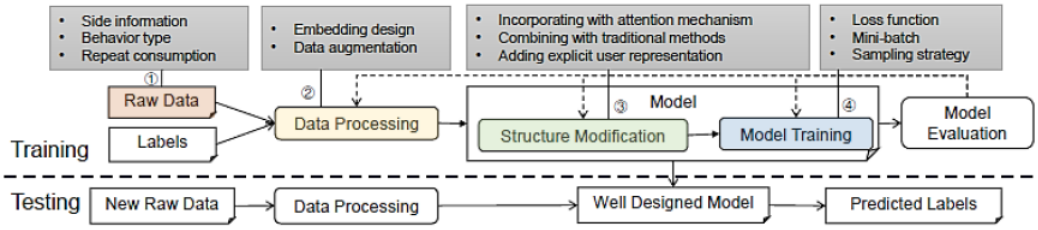


그림 1. 딥러닝 기반 모델에 영향을 미치는 요인들[1]

분류 태스크의 성능을 크게 향상시킨 연구가 주목받으면서 자연어 데이터의 증강에 대한 관심이 높아지고 있다.[5] 순차 추천 시스템 역시 이러한 데이터 증강 기술을 활용하여 추가적인 성능 향상을 꾀할 수 있는 가능성이 존재하나, 아직까지 추천 분야에서 데이터 증강이 얼마나 효과적일 수 있는지에 대해 충분히 검증되지 않았다.

이러한 문제의식을 바탕으로 본 논문에서는 학습 데이터가 충분하지 않은 상황에서 다양한 데이터 증강 기술을 활용함으로써 순차 추천 시스템의 성능을 향상시킬 수 있음을 보이고자 한다. 이를 위해 인공신경망 기반의 최신 순차 추천 모델을 기반으로 원본 아이템 시퀀스를 직접 변형시키는 방식의 총 4 가지 데이터 증강 기법 1)노이즈 추가(Noise Injection), 2)중복성 추가 (Redundancy Injection), 3)아이템 마스킹(Masking), 4)유사 아이템 대체(Synonym Substitution)을 구현하여 추천 성능의 변화를 확인하였다. 순차 추천 모델의 성능 평가를 위한 벤치마크로 널리 사용되는 '무비렌즈 1M(MovieLens-1M)', '고왈라(Gowalla)', '아마존 비디오게임(Amazon Video Games)'의 3 개 데이터셋에 대해 실험을 진행한 결과, 제안하는 데이터 증강 기술을 적용할 경우 데이터셋에 따라 개선 정도에는 차이가 존재하나 전반적인 성능 개선이 나타나는 것으로 확인되었다. 특히 데이터셋의 크기가 작을 때에 데이터 증강에 의한 성능 향상의 폭이 큰 것으로 나타나, 순차 추천 시스템이 적용되는 서비스 초창기 또는 데이터를 충분히 확보하지 못했을 경우 성능 향상을 위해 데이터 증강이 효과적으로 활용될 수 있을 것으로 보인다. 이를 통한 본 연구의 기여 부분을 간단히 정리하자면 다음과 같다.

- 데이터 증강을 통해 학습 데이터가 충분하지 않은 상황에서 순차 추천 성능을 개선할 수 있음을 정량 실험을 통해 확인함
- 데이터 증강은 모델의 아키텍처를 변화시키지 않으므로 현재 제시되어 있는 다양한 SOTA(State-Of-The-Art) 모델의 성능을 추가적으로 개선시킬 수 있는 가능성을 제시함
- 다양한 데이터 증강 기법을 포괄적으로 적용해 봄으로써 데이터 증강이 추천 시스템 디자인에 있어 보편적인 전처리 기법의 하나로 기능할 수 있을 가능성을 검증함
- 향후 순차 추천에서의 데이터 증강 연구에 있어 기초 자료로 활용될 수 있음

제 2 장 선행 연구

본 연구의 내용을 간단하게 정리한다면 딥러닝 기반의 순차 추천 시스템에서 학습 데이터가 충분히 확보되지 않은 경우 다양한 데이터 증강 기술의 활용이 추천 성능 향상에 얼마나 효과적일 수 있는지 확인한 것이라고 할 수 있다. 이에, 이번 장에서는 본 연구가 기반을 두고 있는 1) 개인화 추천 시스템, 2) 순차 추천 시스템, 3) 데이터 증강 각각에 대해 기존에 어떠한 연구들이 이루어져 왔는지 살펴봄으로써 본 연구의 상대적인 위치를 가늠해보고자 한다.

제 1 절 개인화 추천 시스템

딥러닝 이전의 추천 시스템

개인화 추천 시스템은 일반적으로 사용자의 피드백 히스토리(historical feedback) 을 바탕으로 해당 사용자와 아이템 간의 적합성(compatibility) 을 모델링하는 것을 핵심으로 한다. 이 때 사용자의 피드백은 5 점 척도의 점수(rating) 과 같은 명시적인 (explicit) 형태가 될 수도 있고, 또는 단순히 활동 로그(log) 상의 클릭(click), 아이템의 구매(purchase), 구매 후 작성한 리뷰(review) 나 코멘트(comment) 등 해당 피드백이 존재하는지 여부를 통해

선호(preference) 를 확인하는 암시적인(implicit) 형태가 될 수도 있다. 일반적으로 암시적인 형태의 피드백은 명시적 피드백보다 획득하기 용이한 편이나 '관측되지 않은(non-observed)' 데이터의 해석에 있어 단순히 아직 관측되지 않았을 뿐인 것인지, 명확한 불호의 영역에 있는 것인지 확실히 알 수 없다는 점에서 난점이 존재한다.

개인화 추천을 위한 접근법(approach) 은 크게 아이템 자체의 특성(property) 을 활용하는 컨텐츠 기반 필터링 (Contents-Based filtering, CB) 과 아이템 (또는 사용자) 간의 취향(preference) 의 유사성에 기반한 협업 필터링(Collaborative Filtering, CF) 으로 나눌 수 있으며, 특히 협업 필터링 관점이 뛰어난 성능과 확장성을 바탕으로 주요 방법론으로 자리잡아 왔다.[6,7] 기본적인 협업 필터링 (CF) 방법은 세부적으로 각 사용자의 특성을 해당 사용자가 선호하는 아이템들로 나타내고 이들 사용자 간의 유사성을 기반으로 추천을 수행하는 사용자 기반 협업 필터링(User-based CF) 과 반대로 각 아이템의 특성을 해당 아이템을 선호하는 사용자들로 나타냄으로써 아이템 간의 유사성을 기반으로 추천을 수행하는 아이템 기반 협업 필터링(Item-based CF) 로 구분할 수 있다.

한편 행렬분해(Matrix Factorization, *MF*)[8] 기법은 비교적 단순한 구조임에도 뛰어난 성능을 보이며 현재까지도 많은 시스템에서 보편적으로 활용되고 있는 협업 필터링(CF) 기반 접근법의 하나이다. 해당 기법은 사용자-아이템을 각각 행과 열로 하여 특정 사용자의 특정 아이템에 대한 선호를 행렬의 원소 값으로 갖는 행렬에 대해,

일종의 특이값 분해(Singular Value Decomposition, SVD) 를 수행하여 결과적으로 사용자 행렬과 아이템 행렬으로 나누어지도록 분해하는 것이다. 이를 통해 각 사용자 및 각 아이템에 대한 잠재 표현(latent representation), 곧 임베딩 벡터를 획득하게 되며, 사용자 임베딩과 아이템 임베딩 간의 내적을 계산함으로써 해당 사용자의 아이템에 대한 선호도를 예측할 수 있게 된다. 이와 같이 행렬분해(MF) 기법이 구조적으로 별점, 클릭 등 사용자-아이템 간의 상호작용 (interaction) 만을 입력 데이터로 사용 가능한 것과 달리, Factorization Machine(*FM*)[9] 방식은 아이템의 메타데이터(metadata) 와 같은 보조적 특성(side features) 을 직접 입력 데이터의 일부로 추가하여 학습할 수 있다는 점에서 차이가 있다.

한편, 아이템 유사도 모델(Item Similarity Models, ISM) 은 각 사용자에 대해 잠재 요소(latent factors) 형태의 명시적인 모델링을 하지 않고 대신 아이템-아이템 유사도 행렬을 학습한다는 점에서 기존의 협업 필터링 모델과는 다른 결의 접근법을 취하고 있다. *FISM*[10] 모델이 대표적으로 이에 해당하며, 해당 모델은 사용자가 이전에 소비(interact) 한 아이템들과의 유사도를 계산함으로써 특정 아이템에 대한 사용자의 선호도를 예측하게 된다.

컨텐츠 기반 필터링(CB) 방법의 경우 각 아이템이 갖는 고유의 특성만을 활용하므로 협업 필터링(CF) 기법이 갖는 구조적인 한계인 콜드 스타트(Cold-start) 문제가 발생하지 않는다는 점에서 장점이 있다. 그러나 근본적으로 아이템 고유의 특성을 임의로 규정하고 이를

계산 가능한 형태로 표현해내는 것이 어렵다는 문제도 동시에 존재하며, 실제 많은 데이터셋을 통한 실험 결과 컨텐츠 기반 필터링(CB) 기법만을 활용할 경우 추천 성능이 높지 않은 경우가 다수 확인되어 왔다. 이에 현재까지는 컨텐츠 기반 필터링(CB) 기법을 독립적인 모델로 사용하기 보다는 협업 필터링(CF) 기법과의 하이브리드 모델로 구성하여 각 기법의 장단점을 상호 보완하는 방식으로 활용되는 추세이다.[11]

인공신경망 기반의 추천 시스템

최근 수 년 사이 인공신경망(Neural Network) 관련 연구가 다시 주목받기 시작하고 다양한 분야에서 괄목할 만한 성과를 나타냄에 따라, 자연스럽게 추천 분야에서도 인공신경망 구조와 관련 아이디어를 적극적으로 활용해오고 있다.

개인화 추천 시스템에서 인공신경망의 활용은 2006 년에 구독형 동영상 스트리밍 서비스인 넷플릭스(Netflix) 가 자사 서비스의 추천 시스템 성능 향상을 위해 개최한 'Netflix Prize Competition' 에서 제안되었던 제한된 볼츠만 머신(Restricted Bolzmann Machine, RBM)을 활용한 협업 필터링[12] 이 초기의 선구적인 연구로 꼽힌다. 이후 다층 퍼셉트론(Multi-Layer Perceptrion, MLP), 합성곱신경망(Convolutional Neural Network, CNN), 변분 오토인코더(Variational Auto-Encoder, VAE) 등 다양한 심층 인공신경망 구조를 추천 모델에 활용한 연구가 활발히 제시되어 왔다.

고전적인 협업 필터링(CF) 방식에 다층 퍼셉트론(MLP) 구조를 활용한 연구는 *NCF*(Neural Collaborative Filtering)[13] 이 대표적이다. 해당 연구는 기존의 행렬분해(MF) 기법에서 사용자 잠재 표현과 아이템 잠재 표현 간의 상관관계를 학습하는 데에 다층 퍼셉트론(MLP) 네트워크를 사용하였다. 한편 *DeepFM*[14] 는 Factorization Machine(FM) 기법에 다층 퍼셉트론(MLP) 구조를 적용한 것으로서, 이는 FM 을 통해 저차원(low-order) 의 특성 간 관계(feature interactions) 를 학습하는 동시에, 심층 인공신경망을 통해 고차원(high-order) 의 관계를 학습할 수 있다는 것이 특징이다.

개인화 추천에서 합성곱신경망(CNN) 구조를 활용한 연구는 많은 경우에 특성 추출(feature extraction) 을 위해 합성곱신경망 (CNN) 구조를 활용하고 있다. *VPOI*[15] 는 POI(Point-of-Interest) 추천에 있어 시각적 특성(visual features) 가 미치는 영향에 대해 조사하여 이러한 시각 컨텐츠 특성을 활용한 POI 추천 시스템을 제안한 것이다. 또한 문자 형태의(textual) 특성 추출 과정에서 합성곱신경망(CNN) 구조를 활용하여 고차원의 아이템 표현을 모델링하고 이를 확률적 행렬분해(Probabilistic MF) 기법과 결합한 연구로서 *ConvMF*[16] 가 존재한다.

순환신경망(Recurrent Neural Network, RNN) 구조의 경우 시퀀스 데이터를 입력으로 받아 내재한 시계열 패턴(sequential pattern) 을 학습하는 데에 특화되어 있다는 점에서 주로 아이템이 소비된 순서

정보까지도 추가적으로 고려하는 순차 추천(Sequential Recommendation) 연구에서 각광을 받고 집중적으로 활용되어 왔다.

추천 시스템에서 오토인코더(Auto-Encoder, AE) 구조의 활용은 크게 두 가지로, 1) 보다 저차원의 특성 표현(feature representation) 을 학습하는 데에 사용하거나, 2) 디코더(Decoder) 부분을 활용하여 직접 상호작용(interaction) 데이터 행렬의 빈 칸을 메우는 것이다. *AutoRec*[17] 은 기본적으로 협업 필터링(CF) 기법의 연장으로서, 사용자 벡터 또는 아이템 벡터를 오토인코더(AE) 구조의 입력으로 받아 예측값을 포함하는 형태의 벡터로 디코딩(reconstruction) 해내는 것을 목표로 한다. *CDAE* [18] 는 기본적으로 선호 정도(rating) 를 예측하는 것이 아닌 각 아이템의 선호 순위 (ranking) 를 예측하는 알고리즘이라는 점에서 이전의 연구들과 차이가 있다. 해당 모델은 원본 데이터에 가우시안 노이즈(Gaussian Noise) 를 추가한 데이터를 디노이징 오토인코더(Denoising Auto-Encoder, DAE) 구조의 입력으로 받아 처리한다. 각각 변분 오토인코더(VAE) 와 디노이징 오토인코더(DAE) 구조를 활용한 Multi-VAE 와 Multi-DAE[19] 의 경우 모델의 파라미터 예측에 베이지안 추론 기반 접근법을 도입함으로써 CDAE 대비 더 뛰어난 성능을 나타내는 등, 최근에는 변분 오토인코더(VAE) 구조 기반 모델이 다양한 벤치마크 데이터셋에 대해 SOTA 성능을 기록하며 활발하게 연구되고 있다.

제 2 절 순차 추천 시스템

앞서 살펴본 일반적인 추천 시스템에서의 접근법은 기본적으로 어떤 아이템에 대한 사용자의 선호가 고정적(static) 이며, 사용자-아이템 간의 모든 상호작용(interaction) 이 동등한 중요성을 가진다는 가정을 전제로 한다. 이를 바탕으로 사용자의 일반적인 선호(general preference) 를 학습하여 추천을 수행하기 때문에 데이터 (곧, 사용자와 아이템 간의 상호작용) 의 순서는 모델링에 고려할 수 없게 된다. 따라서 시퀀스 데이터 내의 시계열 패턴 (sequential pattern) 을 파악하는 데에 근본적인 한계가 존재하므로, 아이템이 소비된 순서를 고려하는 순차 추천 시스템을 위해서는 별도의 접근법이 요구되어 왔다.[1]

딥러닝 이전의 순차 추천 시스템

딥러닝 이전의 전통적인 방법론에는 대표적으로 1) 패턴 마이닝을 이용한 접근법, 2) K-최근접이웃(Nearest neighbors, KNN) 기반 방식, 3) 행렬 분해(MF) 기반 모델, 4) 마코프 체인(Markov Chain) 기반 모델 등이 존재한다[21].

[1] 최근에는 [20] 와 같이 인공신경망 구조를 협업 필터링에 적용하는 과정에서 시퀀스 정보를 함께 모델링하여 순차 추천에 협업 필터링을 활용한 연구도 제시되고 있다.

패턴 마이닝은 통계적 공출현(co-occurrence) 을 바탕으로 데이터 상에서 명시적인(explicit) 시계열 패턴을 파악하고자 하는 접근법이나, 명시적으로 드러나는 규칙성에 의존할 수 밖에 없으므로 뚜렷하게 드러나지 않은 암시적인(implicit) 패턴은 놓치게 된다는 한계가 있다. K-최근접이웃(KNN) 방식의 경우 아이템 기반의 모델과 세션(session) 기반 모델로 나눌 수 있으며, 각각 아이템 간, 세션 간의 유사성(similarity) 을 구하고 이를 기반으로 추천이 이루어지는 단순하고 직관적인 방법이다. 행렬 분해(MF) 의 경우 일반적인 개인화 추천에도 활용되는 방법론이지만[8], 협업 필터링 관점에서는 사용자-아이템 간 공출현 행렬(user-item co-occurrence matrix) 을 분해(factorize) 하는 데에 비해, 순차 추천 상황에서는 사용자-아이템 간 변화 행렬(user-item transition matrix) 을 분해하여 각각의 잠재 표현을 얻어낸다는 점에서 차이가 존재한다.

마코프 체인(MC) 기반 모델은 초기 접근법 가운데 가장 주류적인 것으로, 시퀀스 내에서 시간에 따른 아이템의 변화 (transition) 를 모델링하는 것이 핵심이다. L-차 마코프 체인 (L-order MC) 은 특정 시점에 대한 추천을 위해 이전 L 개 데이터 (상호작용) 을 고려하며, 모델의 복잡도를 낮추기 위해 인접한 아이템 간의 변화만을 고려하는 1 차 마코프 체인이 주로 활용되어 왔다. 대표적인 마코프 체인 기반 모델의 하나인 *FPMC*[22] 는 이러한 인접 아이템 간 변화 행렬을 두 개의 저차수 잠재 행렬(latent and low-rank matrix) 로 분해함으로써 성능을 개선하였으며, 이어서 등장한 *Fossil*[23] 모델은 해당 접근법을 보다 고차(high-order) 의 마코프 체인으로 확장하였다.

인공신경망 기반의 순차 추천 시스템

순차 추천 시스템의 경우, 다양한 인공신경망 구조 중에서도 입력 데이터를 시퀀스 단위로 받아 모델링 한다는 강력한 공통점을 지닌 순환신경망(RNN) 구조를 활용한 연구가 가장 활발하게 이루어져 왔다. 순환신경망 구조를 활용한 초기 모델 중 가장 대표적인 것으로는 게이트 순환 유닛(Gated Recurrent Units, GRU) 을 이용해 시퀀스 데이터를 모델링하여 랭킹 손실 함수(ranking loss) 로 학습을 진행한 *GRU4Rec*[24] 이 존재하며, 유사하게 *GRU4Rec+*[25] 모델은 해당 모델에서 손실 함수와 샘플링 전략 측면에서 다른 접근법을 사용함으로써 상위 N 개 추천 문제에서 유의미한 성능 향상을 보인 개선 버전이다.

이후 컨텐츠 또는 맥락적 특성을 추가로 반영하거나, 다른 형태의 네트워크 구조를 결합적으로(jointly) 구성하는 등 순환신경망 구조를 기반으로 확장되고 변형된 모델들이 다수 등장하였다. *RCNN*[26] 은 모델의 앞단에서 일반적으로 이용되는 임베딩 계층을 순환신경망 계층으로 대체한 후 이어서 합성곱 신경망(CNN) 구조를 활용함으로써 순환신경망 계층을 통해서는 장기적인(long-term) 시계열 패턴을, 합성곱신경망 계층을 통해서는 상대적으로 단기적인(short-term) 패턴을 종합적으로 파악할 수 있게 한 것이 특징이다. 한편 *VASER*[27] 모델의 경우 게이트 순환 유닛(GRU) 모듈과 어텐션(Attention) 모듈로 구성된 순환신경망 구조에 확률적 잠재 변수(stochastic latent variables) 를 도입하여 세션 생성 모델을 구축하고 변분 추론(Variational

Inference) 방식을 활용하여 다음 아이템을 예측하였다. 순환신경망 구조의 경우 기본적으로 높은 수준의 계산 비용을 요구한다는 점이 한계로 지적되고 있으며, 언어(natural language) 와 같은 시퀀스 데이터를 모델링하는 데에 특화되어 있는 만큼, 인접 아이템 간에 뚜렷한 시계열적 의존 관계가 존재하는 경우 높은 성능을 나타낼 수 있는 것으로 알려져 있다.

기본적으로 순환신경망(RNN) 은 자연어처리(NLP) 분야를 중심으로 적용되고 연구되어 온 구조라는 점에서, 순차 추천 시스템의 모델링에 있어서도 해당 분야의 연구 동향이 적지 않게 반영되어 있으며, 자연어 처리 분야의 문제 해결을 위해 제안된 선도적인 접근법이나 아이디어들을 가져와 순차 추천 문제에 적용해 본 연구가 많다. 이러한 맥락에서 순환신경망에 이어 어텐션(Attention) 메커니즘에 대한 높은 관심과 활발한 적용도 존재해 왔다. 대표적으로 *NARM*[27] 모델은 순환신경망 구조를 통해 인코딩 된 상태들(states)을 결합하는 과정에서 어텐션 레이어를 활용한 것으로서, 이를 통해 모델링 과정에서 입력으로 들어온 시퀀스 데이터에서 어느 부분(아이템)이 더욱 중요한지를 명시적으로 강조할 수 있게 하였다. *MARANK*[28] 모델의 경우 시퀀스 내에서 가장 최신의 일부 아이템만을 대상으로 고차(multi-order) 어텐션을 적용함으로써 개별 아이템 수준 및 아이템 집합 수준에서의 의존 관계(dependency) 를 파악하고자 하였다. *STAMP*[29] 모델의 경우 순환신경망 구조를 구성하는 모든 인코더 대신 이를 어텐션 레이어로 대체하고 시퀀스의 마지막 아이템의 셀프-어텐션(Self-Attention) 에 의지하는 네트워크를 구성하였다. 이를 통해

해당 세션을 관통하는 사용자의 일반적인 선호(interest) 와 마지막 아이템에서 나타나는 직전의 선호를 함께 잡아냄으로써 순환신경망을 이용하는 기존의 연구들과 비교해 보다 강력한 추천 시스템을 만들어 내었다.

한편, 최신의 선행 연구들을 통해 합성곱신경망(CNN) 및 셀프-어텐션(Self-Attention) 이 시퀀스 모델링에 있어서 더 나은 결과를 가져올 수 있음이 확인되면서, 해당 구조들을 활용한 연구들도 다수 등장하였다. 기존의 어텐션 메커니즘이 해당 메커니즘을 구성하는 핵심적인 세 요소인 쿼리(query), 키(key), 밸류(value) 에 대해 키와 밸류만 동일한 아이템으로 간주하는 것과 달리, 셀프-어텐션의 경우 쿼리, 키, 밸류 세 요소를 모두 동일한 아이템으로 간주한다는 점에서 차이가 있다. 자연어처리(NLP) 분야에서 제안된 트랜스포머(*Transformer*)[30] 와 버트(*BERT*)[31] 모델은 이전의 연구들이 어텐션 메커니즘을 여타 네트워크 구조를 보완하기 위한 추가적인 요소로 활용해온 것과 달리, 온전히 셀프-어텐션 메커니즘을 네트워크 구조의 메인 요소로 활용한 것으로, 텍스트 시퀀스 모델링에 있어 단순히 SOTA 성능을 기록한 것을 넘어 연구의 지형을 바꾼 그야말로 혁명적인 연구로 알려져 있다. *SASRec*[32] 은 셀프-어텐션 메커니즘을 순차 추천 연구에서 활용한 최초의 모델로서, 해당 메커니즘을 중심으로 구성된 트랜스포머 모델의 인코더 구조를 활용하여 각 시퀀스를 구성하는 아이템의 임베딩을 학습한다. 이를 바탕으로 아이템 유사도 모델(item similarity models) 의 구조를 활용하여 다음 아이템이 될 후보 아이템들의 점수(score) 를 계산하고 순위(rank) 를 매겨 추천

결과를 도출한다. *SASRec* 은 당시 여러 공개 벤치마크 데이터셋에 대해 SOTA 수준의 성능을 기록하였을 뿐만 아니라 이후 *BERT4Rec*[33], *TiSASRec*[34] 과 같은 현재 SOTA 수준 모델의 기반이 된 연구라고 할 수 있다. *BERT4Rec* 은 자연어 처리 분야의 버트 모델이 텍스트 시퀀스(문장) 를 학습하는 데에 활용 하였던 MLM(Masked Language Model) 태스크를 아이템 시퀀스의 학습에 도입함으로써 마찬가지로 양방향(bi-directional) 방식의 인코딩을 가능하게 한 모델이다. *TiSASRec*[34] 은 앞서 설명한 *SASRec*[32] 모델을 직접적으로 기반으로 하여 해당 모델을 개선시킨 것이다. *SASRec* 이 기존의 일반적인 순차 추천 모델과 마찬가지로 인코딩 과정에서 시퀀스를 구성하는 아이템 간의 '순서(order)' 정보만을 사용하는 방식이라면, *TiSASRec* 모델의 경우 이에 더해 각 아이템이 소비(interact) 되었을 때의 시간 정보(timestamp) 를 사용한 것으로서, 시퀀스 내 아이템 간의 절대적인 시간 간격(time interval) 을 계산하여 이를 추가적인 보조 정보(side information) 으로서 모델링 과정에 활용하였다.

한편, 컴퓨터 비전(Computer Vision) 분야를 중심으로 연구되어 온 합성곱신경망(CNN) 구조의 경우 최근에서야 순차 추천 분야에서 관심을 갖기 시작했으며, 아직 상대적으로 탐구가 충분히 진행되지 않은 접근법이라고 할 수 있다. *CASER*[35] 는 순차 추천 시스템의 성능 향상을 위해 합성곱신경망 구조를 처음으로 도입한 연구라는 점에서 주목할 만하다. 합성곱신경망 구조는 일반적으로 신호(signal) 와 같은 시계열 데이터와 이미지 데이터의 처리에 적용되어 왔으며, 순환신경망 구조 대비 상대적으로 긴 시퀀스에 대해서도 계산 비용이

낮고 데이터에 내재한 국지적 정보들(local information) 간의 의존 관계를 파악하는 데에 적절한 것으로 알려져 있다. *CASER* 는 이러한 합성곱신경망을 이용하여 시퀀스 내 국지적 특성(local features) 을 파악하는 데에 효과적인 모델임이 확인되었다.

일반적으로 순환신경망(RNN) 을 활용한 모델은 시퀀스를 구성하는 아이템 간 시계열적 의존 관계가 뚜렷하게 존재하는 경우 효과적인 것으로 알려져 있으나, 순차 추천 문제의 경우 모든 인접 아이템 사이에 의존 관계가 존재하지는 않을 가능성이 높다. *CASER*[35] 는 인접 아이템 간의 시계열적 의존 관계를 직접 모델링하는 대신, 이전 아이템들의 임베딩 행렬을 하나의 '이미지'로 간주하고 이에 합성곱 필터를 이용하여 시퀀스의 국지적 특성을 잡아내는 방식으로 시퀀스 내의 시계열 패턴을 파악하고자 하였다. 또한 이를 통해, 앞서 설명한 마코프 체인 기반 모델이 지닌 예측 대상이 되는 타겟 아이템에 대해 시퀀스 내 이전 아이템들이 영향을 미치는 시계열 패턴을 개별 아이템 단위로만(point-level) 파악할 수 있다는 한계를 극복하고, 시퀀스 내 아이템 집합 단위(union-level) 의 시계열 패턴과 건너뜀 양상까지 고려함으로써 유의미한 성능 개선을 보여주었다.

이와 같이 *CASER* 모델을 통해 합성곱신경망 구조가 시퀀스 내 단기적/국지적인 시계열 패턴을 파악하는 데에 우수한 성능을 나타낸다는 것이 확인되면서, 해당 연구의 접근법에 착안하여 순차 추천에 합성곱신경망을 활용한 후속 연구가 일부 제안되었다. *CASER* 에서 활용한 기본적인 2D 구조 이외에 3D, 1D 등으로 합성곱신경망 구조 또는 합성곱 필터의 형태를 변형함으로써 추천 성능을 개선하고자

하는 연구들이 차례로 등장하였으며, 이들은 모두 네트워크의 뼈대(backbone) 가 되는 합성곱신경망 구조만을 변형함으로써 성능 개선을 도모하였다는 점에서 공통적인 방향성을 띄고 있다. *NextItNet*[36] 은 이미지 생성 문제의 접근법을 활용하여 합성곱 계층을 팽창된(dilated) 합성곱 필터를 갖는 1D 구조로 변경하고 이를 깊게 쌓는 동시에, 이러한 다층 구조를 보다 효율적으로 학습할 수 있도록 돕는 레지듀얼 블록(residual block) 을 도입하여 기존 모델이 갖는 계산 속도 면에서의 장점을 유지하면서 보다 개선된 성능을 기록하였다. *CosRec*[37] 모델의 경우 이전 모델들이 아이템 시퀀스의 순서를 그대로 살려 네트워크 구조의 입력으로 사용한 것과 달리, 임베딩 계층과 합성곱 계층 사이에 인접한 아이템 쌍들을 대상으로 인코딩(pairwise encoding) 모듈을 추가하여 인접하지 않은(non-adjacent) 아이템 간의 상호작용을 모델링할 수 있게 하였다.

이외에도, 최근 딥러닝 연구 전반에서 주목받고 있는 그래프 인공신경망(Graph Neural Network, GNN) 을 활용한 모델도 순차 추천 분야에서 활용되기 시작하였으며, 이를 통해 그래프 네트워크 구조를 활용하는 것이 우수한 추천 성능을 가져오는 데에 도움이 되는 것으로 확인되었다. *SR-GNN*[38] 는 순차 추천 시스템에 그래프 인공신경망 구조를 적용한 선구적인 연구로서, 순서 정보가 유의미한 특성으로 작용하는 시퀀스 데이터를 모델링하기 위해 방향성이 있는(directed) 그래프 네트워크에 해당하는 게이트 그래프 네트워크(Gated Graph Network) 를 기본적인 모델 아키텍처로 활용하였다. 이 때 게이트 그래프 네트워크를 통해 인코딩된 아이템

임베딩은 셀프-어텐션 계층을 거치며 시퀀스 단위의 특성까지 학습할 수 있게 되는 구조이다. 이어서 등장한 *FGNN*[39] 는 마찬가지로 그래프 인공신경망을 모델의 뼈대로 하고 있으나, *SR-GNN* 모델과 달리 방향성이 없는(undirected) 그래프 네트워크에 해당하는 그래프 어텐션 네트워크(Graph Attention Network, GAT) 를 활용하였을 뿐만 아니라 최종적으로 개별 아이템 단위의 임베딩을 학습하는 것이 아니라 시퀀스(세션) 단위의 임베딩을 학습하여 그래프 분류(graph classification) 문제로 접근하였다는 점에서 명확한 차이가 존재한다. 해당 모델은 대신 추가적으로 *Set2Set*[40] 알고리즘을 활용하여 시퀀스 내 아이템 간 순서 정보를 모델링하고자 하였다.

또한 일반적인 개인화 추천 분야에서 최근 SOTA 성능을 나타낸 변분 오토-인코더(VAE) 를 비롯, 생성 적대 네트워크(GAN) 와 같은 생성 모델(Generative model) 역시 순차 추천 분야에서도 그 성능을 입증하며 주목받기 시작했다.[20]

이처럼 순차 추천 시스템의 성능 향상을 위해 다양한 딥러닝 모델을 활용한 연구가 다수 존재하나, 현재까지 제안된 대다수 접근법은 주로 '모델의 아키텍처 개선' 측면에 집중되어 있다는 한계가 있으며 데이터의 입력, 전처리 및 모델 학습 측면에 초점을 맞추고 있는 연구는 상대적으로 소수에 불과한 것을 확인할 수 있었다.

그러나 딥러닝 기반 모델의 성능에 영향을 미치는 요소는 단순히 모델의 네트워크 구성뿐만 아니라 다양한 요소들이 복합적으로 작용하는 것으로 알려져 있다. 특히 딥러닝 아키텍처를 활용한 모델이 높은 성능을 얻기 위해서는 모델을 구성하는 수많은 파라미터를

추정하기 위한 대규모의 데이터셋이 전제되며, 그렇지 못한 경우 오히려 훨씬 단순한 구조를 가진 고전적인 접근법의 모델보다 더 낮은 성능을 나타낼 가능성도 높다. 이에 본 연구는 기존의 연구들이 상대적으로 적은 관심을 기울여 왔던 데이터 전처리 측면에 주목하고 있다는 점에서 앞서 살펴본 연구들과는 근본적인 차이가 존재한다.

제 3 절 데이터 증강

앞서 서술했듯이 딥러닝 기반 모델이 높은 성능을 얻기 위해서는 모델을 구성하는 수많은 파라미터를 학습하기 위해 충분히 많은 데이터를 필요로 한다. 같은 맥락에서, 딥러닝 모델이 학습 데이터를 넘어 테스트 데이터에 대해서도 일반화 (generalization) 를 잘 하도록 만들기 위한 최선의 방법은 더 많은 데이터를 학습에 사용하는 것이다. 하지만 많은 경우에 현실적으로 활용 가능한 데이터의 양은 제한되어 있으며 이미 확보되어 있는 데이터셋에 더해 추가로 데이터를 구하는 것은 높은 비용을 요구한다. 이에 딥러닝 기법을 활용하는 여러 도메인에서는 기 확보된 데이터를 기반으로 가짜 데이터(fake data) 를 만들어내고 이를 추가적으로 학습에 활용하는 '데이터 증강(Data Augmentation)' 이 대안적인 방법으로서 활용되어 왔다.

일반적으로 딥러닝을 활용하는 다양한 태스크 가운데 '분류(classification)' 문제에 대해서 데이터 증강이 쉽게 적용될 수 있는 것으로 알려져 있다. 딥러닝 모델은 하나의 분류기(classifier) 로서 복잡한 고차원의 데이터를 입력으로 받아 결과적으로 이에 해당하는 하나의 카테고리를 출력하게 되며, 이는 다르게 표현하면 분류기가 입력 데이터의 핵심적인 특성과 무관한 다양한 변형(transformations) 에 대해서는 불변하도록(invariant) 학습 시키는 것이라고 할 수 있을 것이다.[41]

컴퓨터 비전(Compute Vision) 분야에서의 데이터 증강

컴퓨터 비전 분야에서 이러한 데이터 증강이 특히 효과적인 것으로 확인되어 온 태스크는 일종의 분류 문제에 해당하는 '객체 인식(Object Recognition)' 이다. 많은 경우에 이미지 데이터의 경우 이미지를 회전시키거나(rotation), 비율을 조정하거나(scale), 픽셀(pixel) 을 일부 가리는 등의 조정(operation) 을 하더라도 본질적으로 해당 이미지가 나타내는 대상(object) 이 무엇인지 그 의미가 변화하지 않기 때문에, 이런 방식으로 일부 변형이 가해진 데이터를 생성해내어 추가적인 학습 데이터로 활용할 수 있는 것이다. 하지만 데이터셋이 담고 있는 이미지의 특성(숫자, 풍경, 사물 등) 에 따라 사용 가능한 변형 기법에 제한이 존재할 수 밖에 없으며, 이에 해당 분야에서는 널리 활용되는 대규모 벤치마크 데이터셋에 대해 최적의 증강 전략을 직접(manually) 찾아내고자 하는 연구들이 꾸준히 이루어져 왔다.[2]

대표적으로 자연 이미지(natural image) 로 구성된 ImageNet 데이터셋의 경우, [42] 연구에서 제안된 임의 잘라내기(random cropping), 이미지 미러링(mirroring), 색상 변형(coloring shifting) 또는 화이트닝(whitening) 과 같은 데이터 증강 기법을 활용할 경우 성능 향상에 도움이 되는 것으로 확인되었으며, 이는 이후 연구들에서도 조금씩의 변화는 있을지 언정 표준적인 (standard) 전처리 과정으로 자리잡았다. 한편 숫자(digit character) 를 포함하는 MNIST 데이터셋의 경우 우수한 성능을 나타낸 모델들은 주로 왜곡(elastic distortion), 비율 조정(scale), 이동(translation), 회전(rotation) 등의 증강 기법을 적용한 것으로 알려져 있다.[43]

AutoAugment[3] 는 앞서 언급한 연구들과 달리 하나의 특정한 데이터셋에 대한 것이 아니라, 원하는 임의의 데이터셋에 대해 가장 적합한 데이터 증강 기법을 학습을 통해 자동으로 찾는 탐색(search) 모델을 제안하고 있다. 해당 모델은 이동(translation), 회전(rotation)과 같이 일반적으로 사용되는 일련의 증강 기법들과 각각을 적용하는 비율, 규모로 구성된 증강 방식(policy) 의 조합들을 탐색 공간으로 설정하고, 강화학습(Reinforcement Learning) 기반의 탐색 알고리즘을 활용하여 최적의 증강 전략을 찾아준다. 데이터 증강 과정의 자동화를 목표로 하며 이보다 먼저 제안되었던 연구에는 *Smart Augmentation*[44]가 있다. 해당 모델은 동일한 클래스(class) 에 속하는 2 개 이상의 데이터 샘플들을 결합(merge) 하는 방식으로 증강 데이터를 자동으로 생성하는 네트워크이다. 유사하게 *Tran et al.*[45] 의 연구는 학습 데이터셋의 분포를 학습하여 이를 바탕으로 베이지안 접근을 활용하여 증강 데이터를 생성하는 모델을 제안하였으며, *DeVries and Taylor*[46]의 연구에서는 학습된 특성 공간(feature space) 상에서의 변형을 통해 데이터를 증강시키고자 하였다.

한편 최근에는 적대생성모델(Generative Adversarial Network, GAN) 등 생성 모델(Generative model) 을 활용하여 추가적인 데이터를 생성하는 방식의 데이터 증강 접근법도 제안되고 있으며[47], 이들 모델은 기본적으로 네트워크를 통해 증강 데이터를 직접 생성해내는 구조를 취하고 있다는 점이 특징이다.

음성(Speech) 분야에서의 데이터 증강

컴퓨터 비전 분야와 더불어 데이터 증강이 효과적으로 활용되고 있는 또다른 분야는 음성(Speech) 으로서, 특히 음성 인식(Speech Recognition) 태스크를 다루는 과정에서 활발하게 연구되어 왔다. 해당 분야에서는 모델의 입력으로 받는 원본 오디오(raw audio) 에 직접 적용하거나 해당 오디오 신호를 가공하여 생성한 멜스펙트로그램(Mel Spectrogram) 형태의 데이터에 적용 가능한 다양한 변형 기법이 제안되어 왔다.

[48] 은 학습 데이터가 절대적으로 부족한 저자원(low resource) 환경에서 인공적인 데이터를 생성시킴으로써 음성 인식 성능을 개선시키고자 한 선구적인 연구에 해당한다. [49] 연구에서는 깨끗한 오디오 데이터에 잡음 신호(noisy audio signal) 을 주입하는 방식으로 잡음(noisy audio) 를 합성해내어 데이터 증강에 활용하였으며, [50] 에서는 원본 오디오에 속도 변경(Speed perturbation) 을 통한 증강 데이터를 생성해내는 방식으로 대어휘 연속음성인식(Large-Vocabulary Continuous Speech Recognition, LVCSR) 태스크에 있어 성능 개선을 도모하였다. [51] 의 경우 다중 스트림(multi-stream) 음성인식 시스템의 학습 과정에서 특성 드랍아웃(feature dropouts) 을 적용하여 데이터 증강을 수행하였다.

한편, 최근 제안된 *SpecAugment*[4] 는 입력 오디오 원본이 아닌 로그 멜스펙트로그램(Log Mel Spectrogram) 단위로 데이터 증강을 수행하는 기법으로, 로그 멜스펙트로그램을 마치 하나의 이미지로 간주하고 총 3 가지의 변형(deformations) 을 가하는 것이 특징이다. 각각의 변형 방식은 1) 시간 워핑(Time warping), 2) 주파수 마스킹

(Frequency masking), 3) 시간 마스킹(Time masking) 에 해당하며, 이들 변형은 최종적으로 네트워크가 유용한 특성들을 학습하여 더욱 강건한 특성을 갖도록 돕는 것을 목적으로 한다. 해당 기법은 온라인 학습이 가능할 정도로 간단하고 계산(computation) 측면으로도 낮은 비용을 요구하면서도, 훨씬 복잡한 기존 모델들을 성능 면에서 능가하며 보다 효율적인 모델임이 확인되었다.

자연어 처리(NLP) 분야에서의 데이터 증강

일반적으로 복잡한 문법 구조를 갖는 자연어(natural language) 의 경우 데이터의 변형(transformation) 에 있어 일반화된 규칙을 떠올리는 것이 매우 어렵다. 이러한 맥락에서 자연어 처리 분야는 앞서 설명한 컴퓨터 비전, 음성 분야와 비교하면 상대적으로 데이터 증강에 대한 관심이 낮게 형성되어 왔으며, 보편적인(universal) 데이터 증강 기법에 대한 논의가 아직 충분히 이루어지고 있지 않다고 할 수 있다. 하지만 최근 단순한 변형 기법의 적용만으로도 텍스트 분류(Text Classification) 태스크에 대해 큰 폭의 성능 향상을 가져올 수 있음이[5] 확인된 이후로 그 활용 가능성이 새로이 주목받고 있다.

자연어 처리 분야에서 데이터 증강을 활용한 대표적인 연구로는 [52] 가 있으며, 이는 영어로 된 원본 문장을 독일어 문장으로 번역한 후 이를 다시 영어로 번역하는 역-번역(Back-translation) 방식을 통해 변형된 문장을 추가로 생성하고자 하였다. 또다른 연구로는 원본 데이터에 노이즈를 추가하여 증강 데이터를 생성한 [53] 가 있으며, 이들 연구는 공통적으로 기계 번역(machine translation) 태스크에 대해

적용된 것으로서 데이터 증강을 통해 BLEU 점수를 다소 향상시킬 수 있음을 보였다.

텍스트 분류 태스크에 있어서는 [54] 연구가 K-최근접이웃(KNN) 알고리즘을 활용하여 단어 임베딩 간의 유사도를 계산해내고 이를 바탕으로 유의어 대체(Synonym Replacement, SR) 기법을 적용하여 F1-스코어 기준 약 1.4%의 개선을 확인하였다. 한편 최근에 해당 분야에서 큰 주목을 받았던 *EDA*[5] 는 유의어 대체(SR) 를 포함하여 총 4 가지의 데이터 증강 기법을 제안하고 입력 단위에 해당하는 원본 문장에 이들 기법 중 한 가지를 랜덤하게 적용함으로써 텍스트 분류 태스크에 있어 유의미한 성능 개선을 가져올 수 있음을 확인한 연구이다. 해당 연구에서는 문장의 일부 단어를 유사한 단어로 대체하는 유의어 대체(SR) 외에도 한 단어에 대한 유의어를 문장 내 임의의 위치에 삽입하는 임의 삽입(Random Insertion, RI), 문장 내 임의의 두 단어에 대해 서로 위치를 바꾸는 임의 교환(Random Swap, RS), 문장 내 일정 비율의 단어를 임의로 삭제하는 임의 삭제(Random Deletion, RD) 기법을 순환신경망(RNN) 과 합성곱신경망(CNN) 구조를 기반으로 한 모델에 적용하였으며, 총 5 개의 벤치마크 데이터셋에 적용하여 실험을 진행한 결과 특히 작은 규모의 데이터셋에 대해 큰 폭의 성능 향상을 확인하였다.

해당 연구는 구체적인 데이터 증강 기법의 측면에서 시퀀스 형태의 입력 데이터에 일종의 노이즈를 추가하여 직접적인 변형을 가하는 방식을 취하고 있다는 점에서 본 연구와 공통점이 있다. 또한 데이터 증강의 효과를 검증하는 과정에서도, 원본 데이터셋을 더 작은 규모의

부분집합으로 쪼개는 방식으로 저자원(low-resource) 상황을 만들어 내었다는 점에서 방법론적 면에서 본 연구의 주요 기반이 되는 연구라고 할 수 있다.

추천 분야에서의 데이터 증강

위에서 살펴본 바와 같이 그동안 딥러닝을 활용하는 여러 도메인에서 데이터 증강 기법이 활발하게 적용되고 논의되어 왔음에도, 해당 기법이 추천 시스템에 적용된 연구는 아직까지 많지 않다.

AugCF[55] 는 기존의 하이브리드 협업 필터링(hybrid CF) 연구들의 한계를 지적하며 제안된 모델로서, 보조 정보(side information) 을 직접 이용하는 것이 아니라 데이터 증강을 위한 도구로서 활용하는 것이 특징이다. 기존의 하이브리드 협업 필터링 접근법이 희소 데이터(data sparsity) 문제를 완화하기 위해 사용자 프로필(user profile), 아이템 설명(item description, 곧 metadata) 과 같은 보조 정보를 단순히 추가적인 입력 데이터로서 사용하는 방식이었다면, 해당 연구에서는 원본 데이터와 함께 이러한 보조 정보를 조건부 적대생성모델(Conditional GAN, CGAN) 구조의 입력으로 활용하여 유저-아이템 간 상호작용(interaction) 데이터를 추가로 생성해내는 접근법을 취하고 있다. 구체적으로, 조건부 적대생성모델(CGAN) 네트워크는 크게 생성자(generator) 와 구분자(discriminator) 의 두 요소로 구성되며 이들은 2 단계 과정을 통해 학습이 이루어진다. 첫번째 단계에서는 생성자가 최대한 원본과 유사한 데이터 샘플을 만들어내도록 학습이 이루어지고, 두번째 단계에 이르면

생성자는 단순히 증강 데이터 샘플(augmented sample) 을 만들어내는 역할을 맡고, 구분자는 생성자가 만들어낸 샘플이 실제 원본 데이터(real) 인지 증강 데이터(fake) 인지는 더이상 구분하지 않고 해당 데이터의 레이블 정보(like/dislike) 만을 구분해내는 역할을 수행하게 된다. 이러한 접근법을 통해 해당 연구는 데이터가 부족한 (inactive) 사용자에 대해 선별적으로 데이터 증강을 적용함으로써 기존의 접근법 대비 훨씬 효율적인 모델을 만들어 내었다.

Matthias et al.[56] 의 연구는 오프라인 소매점에서의 제품 추천(product recommendation) 상황에 대해 데이터 증강을 적용하고 있다. 해당 연구에서 다루는 오프라인 상황에서의 제품 추천 문제의 경우 제한적인 데이터 활용성(data availability) 과 높은 희소성(sparsity) 으로 인해 특히나 정확한 추천이 어려운 것으로 알려져 있다. 해당 연구에서는 기존에 임베딩 기반 접근법으로서 제안되었던 *prod2vec*[57] 알고리즘을 바탕으로, 판매 장소(Point-of-Sale) 정보를 추가하는 한편 데이터 증강 기법을 적용하여 보다 개선된 모델을 만들어내었다. 제품 추천에 있어서 결국 중요한 것은 '어떤 아이템이 함께 구매되었는지' 에 관한 정보라는 점에서, 해당 연구에서는 이전에 구입된 제품 집합 (previously consumed item set), 곧 하나의 데이터 샘플 크기가 m 일 때, 이를 바탕으로 $m\mathrm{C}2$ ($m \geq 3$ 인 경우 $m\mathrm{C}3$ 까지) 개 조합의 제품 쌍(pair) 을 만들어내는 방식으로 증강 데이터를 생성하였다.

한편, *Yan et al.*[58] 은 패션 추천(fashion recommendation) 문제에 대해 전이 학습(Transfer learning) 방식의 데이터 증강에 더해 지식 그래프(Knowledge graph) 를 활용하여 추천 성능의 개선을 도모한

연구이다. 해당 연구에서는 데이터 증강을 크게 두 가지 접근법으로 구분하고 있는데, 하나는 복잡한 모델을 훈련시키기에 절대적인 데이터 규모가 절대적으로 충분하지 않은 경우에 데이터 수 자체를 늘리는 방법을 가리키고, 다른 하나는 데이터셋의 규모는 크지만 퀄리티가 좋지 않아 데이터셋으로부터 충분한 특성(attributes) 을 얻어내기 어려운 경우 추가적인 전처리를 통해 유의미한 정보를 추출해내는 것을 의미한다. 해당 연구에서 다루고 있는 데이터 증강은 두번째 측면에 초점을 맞춘 것으로, 데이터 필터링(filtering) 과 태깅(tagging) 을 통해 데이터 자체의 퀄리티를 높이고자 하였다. 반면 본 연구에서 다루고자 하는 데이터 증강은 첫번째 접근법에 해당하는 것이라는 점에서 동일한 용어를 사용하고 있으나 의미적으로 다소간 간극이 존재한다고 할 수 있다.

순차 추천 분야에서의 데이터 증강

앞서 살펴본 연구들이 일반적인 추천 문제에 대해 데이터 증강을 적용하고 있다면, 본 파트에서는 특히 순차 추천 문제에서 시퀀스 데이터의 증강을 수행한 연구들에 대해 다루고자 한다. *Improved RNN*[59] 은 길이 m인 아이템 시퀀스로부터 각각 길이 $1\sim(m-1)$인 부분집합, 곧 서브시퀀스(subsequence) 를 만들어내고 각 서브시퀀스에 대해 드랍아웃(dropout) 을 적용하는 방식으로 데이터 증강을 진행하여 기존 *GRU4Rec*[24] 모델의 성능을 크게 개선하였다. 해당 연구는 딥러닝을 활용한 순차 추천 모델링에 있어 데이터 증강을 적용한 최초의 연구인 것으로 확인되며, 이후 해당 분야에서 활용된 데이터

증강 접근법은 모두 이처럼 원본 시퀀스의 부분집합을 활용하는 방식을 따르고 있다. 해당 연구에서의 실험은 RecSys Challenge 2015 를 통해 공개된 벤치마크 데이터셋 YooChoose 에 대해서만 제한적으로 수행되었으며, 이후 해당 연구에서 제안된 데이터 증강 기법은 YooChoose 데이터셋을 활용하는 순차 추천 모델의 경우 기본적인 전처리 과정으로서 적용되어 왔다.[39]

한편 앞서 순차 추천에 합성곱신경망(CNN) 을 활용한 대표 연구로 언급했던 *CASER*[35] 를 데이터 증강의 측면에서의 선행 연구로 볼 수도 있다. 해당 연구는 데이터의 전처리 과정에서 길이 m 인 원본 시퀀스를 동일한 윈도우(길이) L 의 서브시퀀스로 쪼개는 슬라이딩 윈도우(sliding window) 기법을 적용하고 있으며, 해당 기법을 적용할 경우 하나의 원본 시퀀스에서 $(m - L + 1)$ 개의 서브시퀀스가 생성되어 결과적으로 데이터셋 규모가 크게 증가하게 된다.

이들 선행 연구들은 공통적으로 하나의 원본 시퀀스에 대해 순서 정보를 유지하는 서브시퀀스들을 만들어내는 방식으로 데이터 증강을 진행하고 있으나, 본 연구의 경우 이와 달리 원본 아이템 시퀀스에 직접적인 변형(direct corruption) 을 가하는 식으로 증강 데이터를 만들어낸다는 점에서 근본적인 차이가 존재한다. 또한 이러한 접근법의 차이로 인해, 부분집합을 활용하는 기존의 증강 방식들은 증강을 통해 생성할 수 있는 데이터의 개수가 원본 시퀀스의 길이에 의존적인(dependent) 반면, 본 연구에서 제안하는 기법들은 원본 시퀀스 길이와 무관하게 원하는 규모로 증강 데이터를 생성해낼 수 있다는 장점이 있다.

제 3 장 연구 방법 및 설계

앞서 제시한 기존 연구들의 한계점들을 보완하려는 시도로서, 본 논문에서는 기본적인 추천 시스템의 뼈대가 되는 모델 아키텍처를 유지한 채 데이터 전처리 과정에서의 데이터 증강을 통해 순차 추천 시스템의 성능 향상을 가져올 수 있음을 보이고자 한다. 따라서 전반적인 추천 시스템의 설계 및 모델의 구조는 데이터 증강을 적용하기 이전의 베이스라인 모델을 따르게 되며, 본 연구에서는 순차 추천 시스템 모델링에 셀프-어텐션(Self-Attention) 메커니즘을 처음으로 활용한 연구 *SASRec*[32] 를 베이스라인으로 삼고 이에 기반하고 있음을 미리 밝혀둔다.

구체적으로 모델의 구성에 대해 설명하기에 앞서, 본 논문에서 다루고자 하는 순차 추천 문제를 아래와 같이 정의하고자 한다: '사용자' (또는 익명의 세션) 집합 U 와 '아이템' 집합 I 가 존재할 때, 각 사용자 $u \in U$ 에 대해 이전에 소비한 아이템들의 시퀀스 $S^u = (S_1^u, S_2^u, \ldots, S_{|S^u|}^u)$, $S_i^u \in I$ 가 주어진다면, 전체 아이템 집합에서 해당 사용자의 선호와 일치하는 상위-N 개의 적절한 '다음 아이템'을 예측하고자 한다.

제 1 절 데이터 증강 기법

본 연구에서는 딥러닝 기반의 최신 순차 추천 모델을 기반으로 학습 데이터를 아이템 시퀀스 단위에서 증강시키는 4 가지 방식의 데이터 증강 기법을 새로이 제안하며, 이들 각각을 적용하였을 때의 추천 성능의 변화를 1) 데이터 증강을 적용하지 않았을 때, 2) 선행 연구[59] 에서 사용된 데이터 증강 방식을 적용하였을 때와 비교함으로써 그 유용성을 확인하고자 하였다.

순차 추천 분야에서 현재까지 선행 연구들을 통해 제시된 데이터 증강 기법은 원본 데이터의 부분집합, 곧 서브시퀀스(subsequence) 를 증강 데이터로서 사용하는 방식이다. 반면 본 연구에서 제안하는 데이터 증강 기법은 이와 달리 원본 데이터에 직접적인 변형(Direct Corruption) 을 가하여 증강 데이터를 생성해내는 방식에 해당한다. 원본 데이터, 곧 아이템 시퀀스에 일종의 노이즈를 더하여 직접 변형하는 방식은 기존의 선행 연구들에서는 다루어지지 않았던 방식으로, 본 연구에서는 세부적으로 '노이즈 추가(Noise Injection)', '중복성 추가(Redundancy Injection)', '아이템 마스킹(Masking)', '유사 아이템 대체(Synonym Substitution)' 의 네 가지 기법을 제안하고자 한다. 한편, 원본 아이템 시퀀스의 부분집합을 사용하는 방식의 경우 직접적으로 데이터 증강을 통해 추천 모델의 성능 개선을 도모하였던 기존의 선행연구[59] 에서 제시되었던 기법이 해당하며, 본 연구에서는 이에 더해 *CASER*[35] 논문에서 데이터 전처리를 위해 활용된 '슬라이딩 윈도우' 방식을 하나의 독립적인 데이터 증강 기법으로 간주하였다. 본 절에서는 본 연구에서 새롭게 제안하는 총 4 가지 증강 기법 뿐만 아니라 선행 연구들에서 다루어졌던 데이터 증강 기법

가운데 본 실험에서 베이스라인으로서 적용한 2 가지 기법 각각에 대해 구체적으로 서술하고자 한다.

증강 방식	기법	비고
부분집합 사용 (Subset selection)	• 부분집합 분할(Subset Split)	선행 연구에서 제안
	• 슬라이딩 윈도우(Sliding Window)	
직접 변형 (Direct Corruption)	• 노이즈 추가(Noise Injection)	본 연구에서 제안
	• 중복성 추가 (Redundancy Injection)	
	• 아이템 마스킹(Masking)	
	• 유사 아이템 대체 (Synonym Substitution)	

표 1. 증강 방식에 따른 데이터 증강 기법의 분류

데이터 증강 규모 $n_{aug} = k$ 로 하여 증강을 적용할 경우, 각 아이템 시퀀스는 하나의 원본 시퀀스와 최대 $(k-1)$ 개의 증강 데이터 시퀀스로 증대되며, 결과적으로 학습 데이터셋 규모가 최대 k 배로 증가하게 된다. 한편, 본 연구에서는 순수하게 데이터 증강에 의한 성능에의 영향을 확인하기 위해 모든 증강 기법을 적용하는 과정에서 데이터 샘플을 활용하지 않도록 설계하였다. 특히, 본 연구에서 베이스라인으로 삼은 모델이 고정된 길이의 시퀀스를 입력으로 받는 구조를 취하고 있으므로, 원본 데이터 시퀀스가 설정된 최대 길이 m 를 초과하여 존재하더라도 가장 최근의 m 개 아이템만을 사용하도록 하였으며 해당 m 개 아이템으로 구성된 시퀀스에 대해서만 증강 데이터를 생성하도록 하였다.

1.1 노이즈 추가 (Noise Injection)

'노이즈 추가' 방식은 원본 아이템 시퀀스에 포함되지 않는 아이템(negative sample), 곧 노이즈(noise) 를 하나 랜덤 샘플링 하여 시퀀스 내 임의의 인덱스에 삽입하는 것이다.

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
증강 시퀀스	[~~1,~~ 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 9, 13, 14, 15]

표 2. 노이즈 추가 기법 적용 예시

이 때, 모델의 구조 상 증강을 통해 아이템 시퀀스의 길이는 원본 시퀀스의 길이와 동일하게 유지되어야 하므로 노이즈가 추가되는 개수만큼 원본 시퀀스 내의 아이템이 제거되어야 한다. 기존의 연구들에 따르면 특정 아이템 시퀀스의 다음에 올 아이템의 예측에 있어서 가장 최신의 아이템이 중요한 영향을 미치는 한편 시간적으로 멀리 떨어진 아이템은 상대적으로 낮은 영향을 미치는 것으로 알려져 있다. 이를 고려하여 본 연구에서는 시퀀스의 가장 선단에 위치하는 첫번째 아이템을 학습 대상에서 제외하고 나머지에 해당하는 원본 시퀀스의 두번째 이후 데이터에 대해 노이즈를 추가한 것이라고 할 수 있다.

1.2 중복성 추가 (Redundancy Injection)

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
증강 시퀀스	[~~1,~~ 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 9, 13, 14, 15]

표 3. 중복성 추가 기법 적용 예시

'중복성 추가' 방식은 원본 아이템 시퀀스에 포함되어 있는 아이템(positive sample) 을 하나 랜덤 샘플링 하여 시퀀스 내 임의의 인덱스에 삽입하는 것이다.

앞서 서술한 노이즈 추가 방식과 마찬가지로, 중복 아이템을 추가하는 대신 원본 시퀀스에서 상대적으로 적은 영향력을 가지는 첫번째 아이템을 제거함으로써 아이템 시퀀스의 길이를 동일하게 유지하는 방식을 취하였다.

1.3 아이템 마스킹 (Masking)

'아이템 마스킹' 방식은 원본 아이템 시퀀스로부터 임의의 k 개 아이템을 랜덤 샘플링 한 후, 해당 아이템이 마치 없는 것처럼 마스킹 처리하는 방식으로 학습 데이터에서 제외시키는 것이다.

이 때 마스킹 대상이 되는 아이템의 개수는 원본 아이템 시퀀스 길이의 일정 비율로 설정하며, 해당 비율은 모델의 디자인 요소 (design factor) 로서 연구자가 설정 값을 정해주어야 하는 하이퍼 파라미터에 해당하게 된다. 최적의 하이퍼 파라미터 값을 찾아 설정하기 의해 본 연구에서는 마스킹 비율을 전체 아이템 시퀀스의 10% / 20% 으로

설정했을 때, 또는 임의의 하나의 아이템만을 마스킹 했을 때의 세 가지 경우에 대해 무비렌즈(ML-1M) 데이터셋을 기준으로 실험해 보았다. 그 결과 원본 아이템 시퀀스의 20% 에 대해 마스킹을 진행했을 때 가장 성능 개선 효과가 크게 나타나는 것으로 확인하고 해당 비율 값을 나머지 데이터셋에 대해서도 공통적으로 적용하였다.

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
증강 시퀀스	[1, 2, ~~3,~~0, 4, 5, 6, 7, 8, 9, ~~10,~~0, ~~11,~~0, 12, 13, 14, 15]

표 4. 아이템 마스킹 기법 적용 예시

1.4 유사 아이템 대체 (Synonym Substitution)

'유사 아이템 대체' 방식은 원본 아이템 시퀀스로부터 임의의 아이템을 하나 랜덤 샘플링 하여 해당 아이템을 가장 유사한 n 개 아이템으로 대체하는 것이다.

대체 대상이 되는 아이템과 나머지 아이템 간의 유사도를 계산하고 해당 아이템과 가장 유사한 아이템이 무엇인지 판단하기 위해서는 증강 데이터를 생성하기에 앞서 아이템 임베딩을 선제적으로 학습할 필요가 있다. 아이템 임베딩 학습을 위해 다양한 모델이 활용될 수 있으나, 본 연구에서는 구현의 용이성과 계산의 효율성 등을 고려하여 자연어처리(NLP) 분야에서 단어 임베딩 사전 학습을 위해 활발히

사용되는 워드투벡터(Word2Vec) 알고리즘을 통해 사전 학습된 아이템 임베딩을 획득하였다.

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
증강 시퀀스 (m=3)	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ~~11,~~121, 12, 13, 14, 15], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ~~11,~~110, 12, 13, 14, 15], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ~~11,~~101, 12, 13, 14, 15]

표 5. 유사 아이템 대체 기법 적용 예시

이 때, 해당 알고리즘을 통해 학습된 아이템 임베딩이 실제 아이템 간의 유사도를 얼마나 제대로 반영하고 있을지 확인하기 어렵다는 문제가 있다. 그러므로 원본 시퀀스의 아이템을 유사한 아이템으로 대체하는 과정에서 단순히 '가장 유사한' 하나의 아이템으로만 대체하는 것은 증강 데이터의 품질을 저하시킬 우려가 있다. 이에 본 연구에서는 하나의 대체 대상 아이템에 대해 상위 m 개의 유사 아이템으로 대체한 증강 데이터를 만들어내는 방식을 취하였다. 예를 들어, n_{aug} =10 인 경우, 대체 대상이 되는 아이템을 랜덤 샘플링을 통해 3 개 선정한 뒤, 각각에 대해 상위 3 개 유사 아이템으로 대체한 3 개의 증강 데이터를 생성하는 것이다.

1.5 부분집합 분할 (Subset Split)

'부분집합 분할' 방식은 길이 m 인 원본 아이템 시퀀스에 대해 각각 길이 $1 \sim m$ 인 m 개의 서브시퀀스로 분할하여 증강 데이터를 생성하는 것으로, 선행 연구인 [59] 에서 처음으로 제안되었다.

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
증강 시퀀스	[1], [1, 2], [1, 2, 3], ⋯ [1, 2, 3, 4, 5, 6, 7, 8, 9], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

표 6. 부분집합 분할 기법 적용 예시

해당 기법은 구조적으로 원본 시퀀스에 포함된 패딩이 아닌 아이템의 개수에 비례해 증강 데이터 샘플이 만들어진다. 곧, 원본 시퀀스 길이에 따라 생성 가능한 증강 데이터 규모에 제한이 생기는 것으로서, 원본 시퀀스 길이 m 과 데이터 증강 규모 n_{aug} 중 더 작은 수만큼의 증강 데이터를 최대로 생성할 수 있다. 이에 선행 연구에서는 원본 시퀀스 별로 데이터 증강 효과가 지나치게 차이 나는 것을 방지하기 위해 데이터 증강을 통해 생성 가능한 전체 데이터 가운데 가장 최신의 일부 (1/4, 1/64 등) 만을 사용하는 방식을 택하였다.

본 연구에서는 기본적으로 데이터 증강 규모 n_{aug}를 증강 효과에 중요한 영향을 미치는 주요 하이퍼 파라미터로 보고 가급적 모든 데이터 샘플에 대해 동일한 규모로 데이터 증강을 적용하고자 하였으므로 본 기법에 대해서는 n_{aug}를 최대 증강 규모로 간주하였다. 다시 말해, $m > n_{aug}$ 인 경우에는 전체 증강 데이터 중 최신의 n_{aug}개 아이템만을 사용하고, $m > n_{aug}$ 인 경우 m 개만을 사용하였다.

한편, 데이터 증강을 통해 생성된 샘플들 간의 유사성이 매우 높으므로 선행 연구에서는 각 샘플에 대해 드랍아웃(dropout) 을 적용하여 모델의 과적합(overfitting) 을 방지하고자 하였다. 이에 본 연구에서도 선행 연구를 따라 동일한 비율(25%) 로 드랍아웃을 적용하였다.

1.6 슬라이딩 윈도우 (Sliding Window)

'슬라이딩 윈도우' 방식은 길이 m 인 원본 아이템 시퀀스를 크기 L 의 윈도우로 슬라이딩 하며 동일한 크기를 갖는 서브시퀀스로 분할하는 것으로, 선행 연구인 *CASER*[35] 에서 합성곱 신경망(CNN) 구조를 이용하기 위한 데이터 전처리 과정에서 활용되었다.

데이터 증강 방식	예시
원본 시퀀스	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
증강 시퀀스 (L=5)	[1, 2, 3, 4, 5], [2, 3, 4, 5, 6], … [5, 6, 7, 8, 9], [6, 7, 8, 9, 10]

표 7. 슬라이딩 윈도우 기법 적용 예시

해당 기법을 적용할 경우 총 $(m - L + 1)$ 개의 증강 데이터를 생성하게 되며, 선행 연구에서는 윈도우 크기 L 을 5~10 이내의 작은 수로 설정하여 하나의 서브시퀀스 내에 가급적 패딩된 아이템이 없도록 하였다. 한편 해당 기법도 '부분집합 분할' 기법과 마찬가지로 데이터 증강 결과 만들어지는 서브시퀀스의 개수가 원본 시퀀스 길이에 선형적으로 비례하므로 L 을 작게 설정할 경우 데이터 셋에 따라 증강

데이터 셋의 규모가 매우 커질 수 있다. 본 연구에서는 데이터 증강 규모 n_{aug} 를 주요 하이퍼 파라미터로 고려하고 있으므로 각 실험 진행 시 n_{aug} 를 고정하고 원본 아이템 시퀀스 길이 m 에 맞게 윈도우 크기 L 를 조정하는 방식을 취하였다. 한편 본 기법에 대해서도 드랍아웃 기법을 적용하여 과적합을 방지하고자 하였으나, 실제 실험 결과 데이터셋에 따라 드랍아웃 기법을 적용하지 않았을 경우 더 높은 성능을 기록하는 경우가 존재하였다. 이에 따라 드랍아웃 적용 여부(boolean) 역시 하나의 하이퍼 파라미터로 놓고 각 데이터셋에 따라 최적의 값으로 설정하여 모델 학습을 진행하였다.

제 2 절 부분집합 데이터셋 생성

본 연구는 데이터셋의 규모가 충분하지 않은 상황에서 데이터 증강이 순차 추천 모델의 성능 개선에 도움이 될 수 있음을 보이고자 한다. 그러나 '데이터셋의 규모가 충분한지' 여부를 판단하기 위한 절대적인 기준은 현실적으로 존재하지 않는다고 할 수 있으며, 이는 결국 해당 시스템 또는 모델이 활용하고자 하는 데이터셋의 특성에 따라 다를 것이다. 이에, 본 연구에서는 데이터셋의 절대적인 규모(세션 수, 아이템 수, 인터랙션 수 등), 희소 정도(sparsity) 및 평균 시퀀스 길이 등의 측면에서 상이한 특성을 지니는 복수의 데이터셋에 대해 각 데이터 증강 기법들을 적용해 보았다.

이와 함께, 본 연구에서는 벤치마크 데이터셋 각각에 대해 전체의 10~50% 에 해당하는 부분집합 데이터셋을 추가로 생성하여 해당 데이터셋에 대해서도 데이터 증강의 효과를 실험적으로 검증함으로써 '데이터셋의 규모가 충분한지' 를 판단하고자 하였다. 각 부분집합 데이터셋은 원본 데이터셋에서 랜덤 샘플링을 통해 생성 되었다. 따라서 절대적인 데이터셋의 규모만이 줄어들었을 뿐, 희소 정도 및 평균 시퀀스 길이와 같은 특성은 원본 데이터셋의 그것과 크게 차이 나지 않으며 유사한 수준으로 유지되는 것을 확인할 수 있었다. 본 연구에서 사용한 벤치마크 데이터셋에 대한 전반적인 설명은 이후 4 장에서 상술할 것이며, 각 부분집합 데이터셋의 특성을 포함하는 개요표 (description table) 는 부록으로 첨부하였다.

제 3 절 모델 학습

데이터 증강은 데이터 전처리 과정에서 적용되는 기술로서, 이러한 전처리 과정을 거친 데이터를 입력으로 받아 시퀀스를 구성하는 아이템 간의 관계 및 변화 패턴(transition pattern) 을 모델링 하고 최종적으로 추천 대상이 되는 다음 아이템을 예측하기 위해서는 추천 시스템의 뼈대(backbone) 가 되는 모델 아키텍처가 필수적이다. 본 연구에서는 시스템의 아키텍처를 변형 및 고도화 하는 것이 목표로 하고 있지 않으므로 기존의 선행 연구를 통해 유용성이 검증된 대표적인 모델을 가져와서 사용하는 방안을 택하였다.

본 연구에서 시스템의 뼈대이자 성능 개선 효과를 검증하기 위한 베이스라인으로 삼은 연구는 *SASRec*[32] 이다. 해당 연구는 순차 추천에서 셀프-어텐션(Self-Attention) 메커니즘을 활용한 최초의 연구로서, 자연어처리(NLP) 분야에서 셀프-어텐션 메커니즘을 이용하는 방식을 거의 그대로 차용하고 있다고 볼 수 있다. 해당 모델은 셀프-어텐션 메커니즘을 중심으로 구성된 트랜스포머(Transformer)[30] 모델의 인코더(Encoder) 구조를 활용하여 각 시퀀스를 구성하는 아이템의 임베딩을 학습하고, 이를 바탕으로 예측 단계에서는 아이템 유사도 모델(ISM) 구조를 가져와 다음 아이템이 될 후보 아이템들의 점수(score) 를 계산하고 순위(rank) 를 매겨 최종적인 상위 k 개 아이템에 대한 예측, 곧 추천 결과를 도출한다.

한편, 해당 연구는 최근 순차 추천 분야에서 SOTA(State-of-the-art) 성능을 나타낸 대표적인 연구인 *TiSASRec*[34] 의 기반이 된 베이스라인 모델이기도 하다. *SASRec*[32] 이 각 세션을 구성하는 아이템들 사이의 순서 정보, 곧 절대적인 위치(absolute positions) 만을 바탕으로 해당 데이터에 내재한 시계열 패턴을 셀프-어텐션 메커니즘을 이용해 모델링 하는 아키텍처라면 *TiSASRec* 은 이러한 순서 정보에 아이템 간의 상대적인 시간 간격(relative time-intervals)을 추가적으로 고려한 모델이라고 할 수 있는 것이다. 비록 성능 면에서 *TiSASRec* 모델이 좀더 우수한 결과를 내는 것으로 나타났으나, 일반적인 순차 추천 시스템은 순서가 있는 시퀀스 데이터만을 입력으로 받는다는 점에서 해당 모델에서 활용하고 있는 '시간 간격(time-interval)' 정보는 추가적인 보조 정보(side information) 로 간주할 수 있다. 본 연구는 단순히 특정한 모델의 성능을 향상시키는 것을 목표로 하는 것이 아니라, 다양한 딥러닝 기반 순차 추천 모델에의 적용 가능성, 곧 확장성을 하나의 중요한 목표로 삼고 있다. 다시 말해, 딥러닝 기반 순차 추천 시스템의 모델 아키텍처를 변형시키지 않고도 보다 개선된 성능을 이끌어낼 수 있는 일종의 전처리 테크닉으로서 데이터 증강의 효과 및 가능성을 검증하고자 한다는 점에서 보다 보편적인 구조를 갖는 모델 아키텍처를 베이스라인으로 삼을 필요가 있는 것이다. 이러한 점에서 *SASRec* 은 일반적인 순차 추천 모델에서 활용하는 아이템 간 순서 정보만을 활용하고 있으면서도 SOTA 모델에 버금가는 우수한 성능을 나타내는 모델로서 본 연구의 베이스라인으로 선정하기에 적합하다고 판단하였다.

SASRec[32] 모델의 전반적인 구조는 [그림 2] 의 도식을 통해 확인할 수 있다. 해당 모델은 일종의 시퀀스-투-시퀀스(seq2seq) 구조로서, 학습 과정에서는 시퀀스의 마지막 아이템 $S^u_{|S^u|}$ 을 제외한 $(S^u_1, S^u_2, \ldots, S^u_{|S^u|-1})$ 을 입력으로 받아 이를 평행 이동한(shifted) 형태인 $(S^u_2, S^u_3, \ldots, S^u_{|S^u|})$ 을 타겟 시퀀스(target sequence) 로서 출력하게 된다. 해당 모델 아키텍처는 본 연구에서 새로이 기여하는 부분이 아니므로 전체적인 구성과 역할에 대해 간단하게 설명한다.

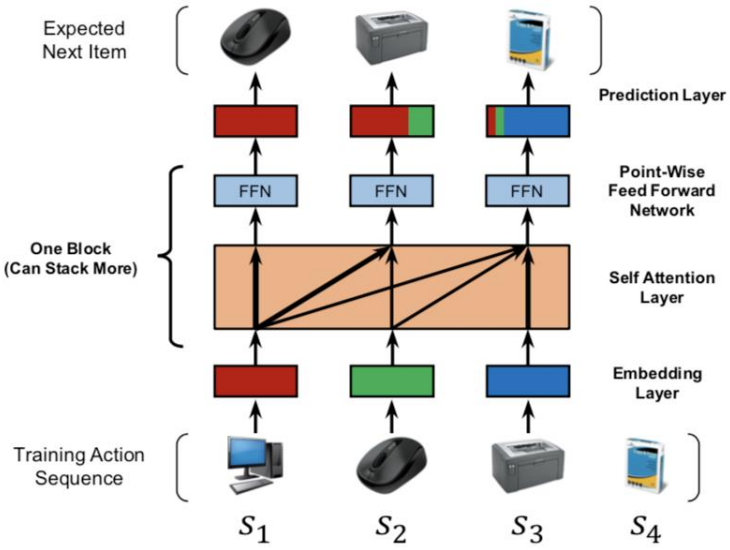


그림 2. *SASRec* 모델의 전반적인 구조[32]

SASRec 모델 구성 및 학습 과정

모델의 아키텍처는 크게 1) 임베딩 계층(Embedding layer), 2) 셀프-어텐션 계층(Self-Attention layer), 3) 예측 계층(Prediction layer)으로 구성된다. 원본 아이템 시퀀스 $(S^u_1, S^u_2, \ldots, S^u_{|S^u|-1})$ 는 임베딩 계층으로 입력되기에 앞서 모델이 받아들일 수 있는 구조와 형태로의 전처리를 거치게 된다. 각 원본 아이템 시퀀스는 다양한 개수의

아이템으로 구성되어 있을 것이나, 해당 모델은 구조적으로 고정된(fixed) 길이의 시퀀스 데이터를 입력으로 받아 처리하도록 되어있다. 따라서 모든 아이템 시퀀스는 하이퍼 파라미터로서 미리 지정된 최대 시퀀스 길이 n 를 갖도록 패딩(padding) 아이템을 추가하거나 가장 최근의 n 개 아이템만 남기는 방식으로 변형되어 $s = (s_1, s_2, \dots, s_n)$ 가 된다. 본 연구는 해당 과정 이후에 $s = (s_1, s_2, \dots, s_n)$를 입력으로 받아 N_{aug}만큼의 증강 데이터를 생성해내는 모듈을 추가한 것이며, 따라서 모델 아키텍처 차원에서의 변형은 이루어지지 않는다.

임베딩 계층에서는 각 아이템에 대한 d-차원의 임베딩으로 구성된 임베딩 행렬 $M \in R^{|I| \times d}$ 과 함께 원본 시퀀스의 순서 정보를 잃어버리게 되는 셀프-어텐션 메커니즘의 한계를 보완하기 위해 추가적으로 포지셔널 임베딩 $P \in R^{n \times d}$ 에 대한 학습이 이루어진다. 이때 포지셔널 임베딩은 이전의 *Transformer*[30] 모델에서 활용한 것과 같이 시퀀스 내 각 위치에 대해 고정된 임베딩 값을 부여하는 방식이 아닌, 별도의 파라미터를 가지며 학습 가능한(learnable) 임베딩을 사용함으로써 보다 나은 성능을 기록할 수 있었다고 한다. 모델의 학습 과정에서 아이템 임베딩과 포지셔널 임베딩이 각각 학습되면 입력 데이터 시퀀스는 임베딩 계층을 거치며 입력 임베딩 행렬(input embedding matrix) $\hat{E} \in R^{n \times d}$ 형태로 변환된다.

$$\hat{E} = \begin{bmatrix} M_{s_1} + P_1 \\ M_{s_2} + P_2 \\ \cdots \\ M_{s_n} + P_n \end{bmatrix} \quad \begin{matrix} (M_{s_n} \in R^d : n\text{번째 아이템 } s_n\text{의 아이템 임베딩}) \\ (P_n \in R^d : n\text{번째 아이템의 포지셔널 임베딩}) \end{matrix}$$

곧, 각 시퀀스를 구성하는 하나의 아이템에 대한 최종적인 임베딩 $\hat{E}_{s_n}$은 아이템 임베딩 M_{s_n} 과 포지셔널 임베딩 P_n 을 결합한 것으로서 획득된다.

셀프-어텐션 계층은 셀프-어텐션 블록(Self-Attention block) 과 포인트 단위 피드포워드 네트워크(Point-Wise Feed-Forward Network)으로 구성된다. 이 때 어텐션 점수는 *Transformer*[30] 모델과 동일하게 스케일드 닷-프로덕트(Scaled dot-product) 방식으로 계산되었다. 스케일드 닷-프로덕트 방식은 쿼리(query) Q, 키(key) K, 밸류(value) V 간의 관계를 아래와 같은 수식으로 계산해낸 것으로서, 아이템 i (쿼리) 와 j (키) 간의 상호작용(interaction) 을 바탕으로 아이템 i (쿼리) 와 j (밸류) 간의 가중치(weight) 를 계산하여 모든 밸류에 대한 가중합 (weighted sum) 을 구한 것이다. 참고로 해당 모델은 셀프-어텐션 메커니즘을 사용하고 있으므로 이 때 쿼리 Q, 키 K, 밸류 V 는 모두 동일한 아이템을 가리키게 된다.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V},$$

수식 1. 스케일드 닷-프로덕트 어텐션의 정의

앞서 임베딩 계층을 거쳐 획득한 아이템 임베딩 행렬 $\hat{E}$ 는 각각 다른 가중치 행렬 $W^Q, W^K, W^V \in R^{n \times d}$ 을 통한 선형 프로젝션(linear projection) 을 거쳐 셀프-어텐션 블록의 입력으로 주어지게 된다. 이 때, $(t+1)$ 번째 아이템을 예측하는 데에 있어 선행하는 t 개 아이템에

대한 정보만을 활용하도록 i > j 인 경우 Q_i 와 K_j 간의 모든 연결을 차단하여 인과성(causality) 을 확보하게 하였다.

$$\mathbf{S} = \mathrm{SA}(\widehat{\mathbf{E}}) = \mathrm{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V),$$

수식 2. 셀프-어텐션 블록에서의 계산

이렇게 셀프-어텐션 블록을 통해 모든 이전 아이템에 대한 통합(aggregation) 이 이루어지면 해당 결과는 포인트 단위 피드포워드 네트워크를 거치며 비선형성(non-linearity) 을 추가로 학습하게 된다.

$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}(\mathbf{S}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

수식 3. 피드포워드 네트워크에서의 계산

해당 모델에서 셀프-어텐션 블록은 복수 개를 쌓음으로써 (stack) 보다 복잡한 아이템 변환(item transition) 관계를 학습하게 할 수 있으며, 본 실험에서는 공개된 베이스라인 모델 코드의 기본값(default) 을 따라 블록 개수를 2 개로 설정하였다. 또한 복수의 블록을 사용할 경우 네트워크가 깊어짐에 따라 모델의 과적합(overfitting) 및 그래디언트 소실(vanishing gradient) 문제가 발생할 수 있으므로, 베이스라인 모델과 동일하게 드랍아웃 (dropout) 과 레지듀얼 연결(residual connection), 계층 정규화 (layer normalization) 을 수행하여 이를 방지하도록 하였다.

모델의 순전파(forward) 과정에서는 전처리를 거친 아이템 시퀀스 $s = (s_1, s_2, \dots, s_n)$ 가 입력으로 주어졌을 때 정답(positive sample, 곧

target) 이 되는 $s' = (s_2, s_3, \dots, s_{n+1})$ 을 예측하는 태스크를 수행한다. 이 때 각 입력 시퀀스에 대해 정답 시퀀스와 페어(pair) 가 되는 오답 시퀀스(negative sample) 를 샘플링 하여 이진 교차 엔트로피 손실(Binary Cross-Entropy Loss) 를 최소화 하도록 모델을 적합(fit) 시킨다.

학습된 모델의 성능 평가를 위한 예측 과정에서는 b 개의 셀프-어텐션 블록을 거치며 추출된 이전 t 개 아이템들에 대한 정보, 곧 $F_t^{(b)}$ 를 바탕으로 그 다음에 올 아이템 $s_{(t+1)}$ 에 대한 예측이 이루어진다. 구체적으로, 예측 계층에서 아이템 i 가 다음 아이템 $s_{(t+1)}$ 에 해당할 연관성(relevance) 을 가리키는 $r_{i,t}$ 의 계산은 행렬분해(MF) 기법을

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{M}_i^T.$$

수식 4. 예측 계층에서의 계산
($r_{i,t}$ = 아이템 i 가 다음 아이템 $s_{(t+1)}$ 에 해당할 연관성)

활용하여 이루어진다. 이 때 아이템 i 에 대한 임베딩 M_i^T 은 임베딩 계층에서 학습한 임베딩 행렬 M 에서 가져온 것으로서, 이러한 아이템 임베딩 공유(share) 를 통해 모델의 규모를 줄이는 한편 과적합을 방지하고자 하였다. 이렇게 계산된 $r_{i,t}$ 는 곧 다음 아이템이 될 후보로서 각 아이템 i 가 갖는 점수(score) 라고 할 수 있으며, 이를 바탕으로 최종적인 상위 N 개 아이템 추천을 위한 순위(rank) 가 매겨지게 된다.

이 때, 해당 모델에서는 효율적인 성능 평가를 위해 아이템 집합 I 에 포함되는 모든 아이템에 대해 순위를 매기는 것이 아니라, 정답(ground-truth) 에 해당하는 아이템 하나와 랜덤 샘플링 된

100 개의 오답(negative items) 으로 구성된 총 101 개 아이템에 대해서만 예측 순위를 매겨 성능을 평가하였다.

제 4 장 실 험

이번 장에서는 본 연구에서 진행한 오프라인 정량 실험 내용과 그 준비 사항에 대해 서술한 후, 그 결과에 대해 구체적으로 설명한다. 본 연구에서 실험을 통해 답하고자 했던 연구 질문은 다음과 같다:

RQ1: 본 연구에서 제안하는 데이터 증강 기법들이 순차 추천에서의 성능 향상을 가져오는가?

RQ2: 데이터 증강 방식에 따라 동일한 데이터셋에 대해서도 데이터 증강의 효과가 다르게 나타나는가?

RQ3: 데이터 증강 규모에 따라 동일한 데이터셋에 대해서도 데이터 증강의 효과가 다르게 나타나는가?

제 1 절 데이터셋 및 실험 셋업

본 연구의 주요 목표 중 하나가 상이한 특성을 갖는 여러 데이터셋에 대해 데이터 증강의 효과를 검증하는 것이므로, 본 실험에서는 영화, 관심 장소(Point of Interest, POI), 이커머스(E-Commerce) 등을 포함하는 다양한 도메인의 데이터셋에 대해 실험을 수행하였다. 본 실험에서 사용된 데이터셋은 1) 무비렌즈 (MovieLens 1M, 이하 ML-1M), 2) 고왈라(Gowalla), 3) 아마존 비디오 게임

데이터 셋	세션 수 (사용자 수)	아이템 수	전체 인터랙션 수	시퀀스 평균 길이[2]	희소 정도 (sparsity)
무비렌즈 (ML-1M)	6,040	3,416	999,611	165.50	95.16
고왈라(Gowalla)	85,034	308,957	6,442,892	52.83	99.98
아마존 비디오 게임(Games)	64,073	33,614	568,508	6.88	99.97

표 8. 사용된 데이터셋에 대한 기술 통계 (전체 데이터셋 기준)

(Amazon Video Games, 이하 Games) 의 총 3 개로, 각각에 대한 특성 통계치는 아래의 [표 8] 를 통해 확인할 수 있다.

무비렌즈(MovieLens) 데이터셋은 추천 시스템 연구 전반에서 광범위하게 사용되고 있는 대표적인 벤치마크 데이터셋으로서 영화에 대한 사용자들의 평가(rating) 로 구성되어 있다. 본 연구는 전체 데이터셋에서 1 백만 개의 사용자 인터랙션(interaction) 정보만을 포함하는 무비렌즈 1M(ML-1M) 버전을 사용하였다. 해당 데이터셋은 여타 벤치마크 데이터셋과 비교해 희소 정도(sparsity) 가 낮으며 평균 시퀀스 길이도 긴 편으로 상대적으로 조밀한(dense) 특성을 갖는 데이터셋으로 간주된다.

고왈라(Gowalla) 는 위치 기반 소셜 네트워킹 서비스 (SNS) 의 이름으로, 동명의 데이터셋은 해당 웹사이트를 이용한 사용자들이 체크인(check-in) 을 통해 자신의 위치 정보(POI) 를 공유한 데이터를 포함하고 있다. 해당 데이터셋은 절대적인 규모가 크고 평균 시퀀스

[2] 세션 당 평균 아이템 수

길이도 50 내외로 짧지 않은 편이나, 희소 정도가 99.98% 로 상당히 높은 것이 특징이다.

한편 아마존(Amazon) 데이터셋 시리즈는 글로벌 이커머스 플랫폼 아마존닷컴(*Amazon.com*) 에서 대규모로 수집된 사용자들의 제품 리뷰로 구성되어 있으며, 상위 제품군에 따라 별도의 데이터셋이 기본적인 포맷을 공유하며 구축되어 있다. 다양한 제품군 가운데 본 연구에서는 '비디오 게임(Games)' 에 해당하는 데이터셋을 사용하였으며, 해당 데이터셋은 높은 희소 정도와 다양성(variability) 이 특징인 것으로 알려져 있다.

데이터셋의 전처리는 기본적으로 선행 베이스라인 연구인 *SASRec*[32] 과 동일한 방식으로 진행하였다. 사용자의 아이템에 대한 리뷰 또는 평가가 존재하는 경우 해당 시점(timestamp) 에 아이템을 소비한 것으로 간주하여 구체적인 평가 내용은 버리고 소비 여부만을 암시적 피드백(implicit feedback) 으로 활용하였다. 모든 데이터셋에 대해 최소 등장 횟수가 5 미만인 사용자나 아이템은 모델 학습을 위한 충분한 정보가 제공되지 않는다고 판단하여 제거하였다. 하나의 세션, 곧 각 아이템 시퀀스 내를 구성하는 데이터(interaction) 들은 시점을 나타내는 타임스탬프 (timestamp) 정보를 기준으로 시간 순 정렬되었다.

3 개 데이터셋 모두 각 시퀀스에 대해 가장 최근에 소비된 아이템을 모델의 성능 평가를 위한 시험 데이터(test set) 로, 그 이전에 소비된 아이템을 하이퍼 파라미터 튜닝을 위한 검증 데이터(validation set) 로, 나머지 아이템들을 학습 데이터(training set) 으로 삼고 이에 맞게 데이터를 분할(split) 하였다. 참고로 성능 평가 단계에서는 기존의

학습 데이터와 함께 검증 데이터를 포함한 데이터로 학습하여 성능을 측정하였다.

본 연구에서는 하나의 추천 모델을 기준으로 선행 연구에서 제시된 기법을 포함하는 다양한 데이터 증강 기법을 적용하여 성능을 확인하고자 하였으므로, 기준이 되는 추천 시스템의 아키텍처는 앞서 상술한 *SASRec*[32] 하나에 대해서만 실험을 진행하였다. 다만, 데이터 증강 기법을 적용하지 않은 바닐라 *SASRec* 모델 외에, 해당 바닐라 모델에 선행 연구들에서 제시된 데이터 증강 기법을 적용한 모델을 추가적인 베이스라인 모델로 삼고 본 연구에서 새로이 제안하는 데이터 증강 기법을 적용한 모델을 이들 복수의 베이스라인 모델과 비교함으로써 그 유용성을 검증하고자 하였다.

베이스라인 모델의 경우 논문 원저자에 의해 텐서플로우(Tensorflow) 1.12 버전 및 파이썬(Python) 2 를 기반으로 구현되어 깃헙(Github) [3] 에 공개되어 있으며, 본 실험에서는 해당 코드를 참고하여 파이토치(Pytorch) 1.6 기반으로 리팩토링한 코드 [4] 를 활용하였다. 또한 데이터 증강을 적용한 모델의 구현에 있어, 데이터 증강 효과의 검증과 무관하게 기본적인 모델 아키텍처와 관련된 하이퍼 파라미터는 기본적으로 베이스라인 모델에서 제시된 최적값(optimal)과 동일하게 설정하였다. 모든 데이터셋에 대해 공통적으로 2 개의 셀프-어텐션 블록과 (b=2) 순서 정보를 위해 학습된 포지셔널 임베딩(learned positional embedding) 이 사용되었으며, 학습률(learning

[3] https://github.com/kang205/SASRec
[4] https://github.com/pmixer/SASRec.pytorch

rate) 는 0.001 로 설정하였다. 배치 크기와 드랍아웃 비율은 데이터셋의 규모와 희소 정도를 고려하여 다르게 설정하였다. 상대적으로 작은 규모인 무비렌즈(ML-1M) 원본 데이터셋은 각각 128, 0.2 로 설정하되, 데이터 증강을 통해 규모가 n_{aug}배로 증가하는 것을 고려하여 증강이 적용된 데이터셋에 대해서는 각각 512, 0.5 로 설정하여 실험하였다. 규모가 크고 희소 정도도 높은 고왈라(Gowalla) 및 아마존 비디오 게임(Games) 데이터셋의 경우 배치 크기를 512, 드랍아웃 비율을 0.5 로 설정하여 실험을 진행하였다. 각 아이템 시퀀스의 최대 길이는 해당 데이터셋의 평균 시퀀스 길이에 따라 무비렌즈(ML-1M), 고왈라(Gowalla) 에 대해서는 50, 아마존 비디오 게임(Games) 에 대해서는 25 으로 설정하였다. 데이터 증강 규모 n_{aug} 는 10 을 기본값으로 설정하여 실험을 진행하였으며, 해당 하이퍼 파라미터의 영향력을 확인하기 위해 $n_{aug} \in \{2,5,15\}$ 에 대해서도 추가 실험을 진행하되 이는 각 방식을 대표하는 일부 증강 기법들에 대해서만 적용해보고 성능 변화를 확인하였다.

결과적으로 본 연구에서 진행한 실험은 3 개 데이터셋 × 6 개 부분집합 × 6 가지 증강 기법 × 4 가지 증강 규모(일부) 의 구조를 갖고 있다고 정리할 수 있다.

제 2 절 성능 평가 지표

모델의 성능은 선행 연구에서 사용된 것과 동일하게 일반적으로 상위 K 개 예측의 성능 평가에 활용되는 다음 두 가지 지표(metric) 를 사용하여 평가하였다: HR(Hit Ratio), NDCG (Normalized Discounted Cumulative Gain).

Hit Ratio 는 단순히 모델이 예측한 상위 K 개 아이템 중에 정답이 존재하는지 여부만을 반영하는 평가 방식으로, HR@K 의 값은 정답(ground-truth) 에 해당하는 다음 아이템(next item) 중 모델이 상위 K 개 아이템으로 예측해낸 비율을 의미한다. 본 실험에서는 각 세션(시퀀스) 에 대해 정답에 해당하는 검증 데이터(validation set) 및 평가 데이터(test set) 가 각각 하나 뿐이므로 재현율 (Recall@K) 과 동일한 값을 갖게 된다.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$iDCG_p = \sum_{i=1}^{\{REL_p\}} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$NDCG_p = \frac{DCG_p}{iDCG_p}$$

수식 5. NDCG 지표의 계산 방식

한편 NDCG 는 예측 순위를 추가적으로 고려하는 평가 지표로서, 정답이 더 높은 순위로 예측되었을 경우에 더 큰 가중치를 부여하는

방식이다. 해당 아이템이 예측된 순위를 i 라 하고 $i \leq \mathrm{K}$ 로서 상위 K 개 아이템 중 하나로 예측되었더라도, $\log_2(i+1)$ 를 분모로 하므로 상대적으로 더 순위에 해당할수록 큰 페널티를 부여하게 된다.

한편, 베이스라인 모델의 경우 상위 10 개 아이템 기준의 (N=10) 성능만을 확인한 것과 달리, 본 연구에서는 N=5 기준의 성능도 추가적으로 확인하였다. 각 모델의 성능은 5 개 랜덤 시드(seed) 에 대한 결과의 평균 값으로 기록하였으며, 해당 결과의 최고 성능(best performance) 은 순위 정보까지 추가적으로 고려한 NDCG@10 을 기준으로 결정하였다.

제 3 절 실험 결과

본 절에서는 기본적으로 무비렌즈(ML-1M), 고왈라(Gowalla), 아마존 비디오게임(Games) 데이터셋 각각에 대해 데이터 증강을 적용하였을 때의 성능 평가 결과에 대해 서술한 후, 이들 결과를 바탕으로 앞서 서술한 연구 질문에 답하고자 한다. 본 연구에서 제안하는 데이터 증강 기법의 효과는 1) 아무런 데이터 증강을 하지 않은 경우(베이스라인), 2) 선행 연구에서 제안한 데이터 증강 기법을 적용한 경우의 두 가지 베이스라인에 대해 비교함으로써 확인하고자 하였다. 각 데이터셋 별 실험 결과를 상술하기에 앞서 주요 실험 결과를 요약하면 다음과 같다.

- 제안하는 데이터 증강 기법들을 적용할 경우, 대부분의 부분집합 데이터셋에서 성능 향상이 확인됨
- 동일한 데이터셋이라도 적용되는 증강 기법에 따라 성능 변화가 다르게 나타나나, 전반적으로 '추가(Injection) 계열' 및 '슬라이딩 윈도우' 가 가장 효과적임
- 모든 데이터셋에 대해 성능 향상 폭은 데이터 증강 규모와 비례하는 경향이 나타남

무비렌즈(ML-1M) 데이터셋 실험 결과

무비렌즈(ML-1M) 데이터셋에 대한 성능 평가 결과는 아래의 [표 9-1~9-6] 를 통해 확인할 수 있다.

ML1M_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.2518 ±.0156		.3114 ±.0118		.3705 ±.0149		.5493 ±.0125	
선행	분할	.3248 ±.0110	(+29.0%)	.3786 ±.0086	(+21.6%)	.4685 ±.0111	(+26.5%)	.6335 ±.0083	(+15.3%)
	슬라이딩	.3401 ±.0091	(+35.1%)	.3936 ±.0045	(+26.4%)	.4821 ±.0143	(+30.1%)	.6473 ±.0119	(+17.8%)
제안	노이즈	.3381 ±.0071	(+34.3%)	.3902 ±.0038	(+25.3%)	.4818 ±.0134	(+30.0%)	.6427 ±.0160	(+17.0%)
	중복성	.3491 ±.0114	(+38.7%)	.4003 ±.0098	(+28.5%)	.4924 ±.0091	(+32.9%)	.6507 ±.0061	(+18.5%)
	마스킹	.3437 ±.0089	(+36.5%)	.3955 ±.0072	(+27.0%)	.4877 ±.0053	(+31.6%)	.6483 ±.0101	(+18.0%)
	유사대체	.3283 ±.0069	(+30.4%)	.3822 ±.0053	(+22.8%)	.4689 ±.0082	(+26.6%)	.6358 ±.0116	(+15.7%)

표 9-1. 무비렌즈 10% 부분집합 기준 성능 평가

ML1M_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3791 ±.0134		.4305 ±.0104		.5270 ±.0151		.6858 ±.0069	
선행	분할	.4001 ±.0108	(+5.5%)	.4434 ±.0095	(+3.0%)	.5614 ±.0160	(+6.5%)	.6947 ±.0098	(+1.3%)
	슬라이딩	.4125 ±.0065	(+8.8%)	.4576 ±.0038	(+6.3%)	.5659 ±.0152	(+7.4%)	.7050 ±.0089	(+2.8%)
제안	노이즈	.4088 ±.0044	(+7.8%)	.4536 ±.0042	(+5.4%)	.5639 ±.0033	(+7.0%)	.7025 ±.0.39	(+2.4%)
	중복성	.4157 ±.0055	(+9.6%)	.4609 ±.0040	(+7.1%)	.5679 ±.0106	(+7.8%)	.7066 ±.0079	(+3.0%)
	마스킹	.4037 ±.0122	(+6.5%)	.4510 ±.0100	(+4.8%)	.5576 ±.0101	(+5.8%)	.7038 ±.0039	(+2.6%)
	유사대체	.4063 ±.0076	(+7.2%)	.4532 ±.0061	(+5.3%)	.5607 ±.0039	(+6.4%)	.7055 ±.0069	(+2.9%)

표 9-2. 무비렌즈 20% 부분집합 기준 성능 평가

ML1M_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4454 ±.0066		.4876 ±.0053		.6041 ±.0097		.7340 ±.0082	
선행	분할	.4381 ±.0054	(−1.6%)	.4805 ±.0044	(−1.5%)	.5950 ±.0055	(−1.5%)	.7262 ±.0077	(−1.1%)
	슬라이딩	.4469 ±.0071	(+0.3%)	.4903 ±.0057	(+0.5%)	.6000 ±.0091	(−0.7%)	.7336 ±.0041	(−0.1%)
제안	노이즈	.4576 ±.0029	(+2.8%)	.5012 ±.0035	(+2.8%)	.6116 ±.0025	(+1.3%)	.7462 ±.0066	(+1.7%)
	중복성	.4551 ±.0068	(+2.2%)	.4974 ±.0050	(+2.5%)	.6116 ±.0107	(+1.2%)	.7418 ±.0047	(+1.1%)
	마스킹	.4504 ±.0075	(+1.1%)	.4937 ±.0051	(+1.2%)	.6084 ±.0063	(+0.7%)	.7414 ±.0064	(+1.0%)
	유사대체	.4523 ±.0024	(+1.6%)	.4955 ±.0012	(+1.6%)	.6066 ±.0043	(+0.4%)	.7393 ±.0027	(+0.7%)

표 9-3. 무비렌즈 30% 부분집합 기준 성능 평가

ML1M_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4671 ±.0085		.5090 ±.0086		.6276 ±.0081		.7566 ±.0078	
선행	분할	.4579 ±.0046	(−2.0%)	.5016 ±.0042	(−1.5%)	.6184 ±.0037	(−1.5%)	.7527 ±.0050	(−0.5%)
	슬라이딩	.4751 ±.0022	(+1.7%)	.5155 ±.0023	(+1.3%)	.6371 ±.0013	(+1.5%)	.7611 ±.0019	(+0.6%)
제안	노이즈	.4773 ±.0042	(+2.2%)	.5179 ±.0027	(+1.8%)	.6387 ±.0043	(+1.8%)	.7635 ±.0030	(+0.9%)
	중복성	.4812 ±.0038	(+3.0%)	.5215 ±.0047	(+2.5%)	.6432 ±.0052	(+2.5%)	.7677 ±.0036	(+1.5%)
	마스킹	.4758 ±.0030	(+1.9%)	.5165 ±.0021	(+1.5%)	.6400 ±.0038	(+2.0%)	.7654 ±.0043	(+1.2%)
	유사대체	.4781 ±.0044	(+2.4%)	.5189 ±.0030	(+2.0%)	.6412 ±.0064	(+2.2%)	.7671 ±.0057	(+1.4%)

표 9-4. 무비렌즈 40% 부분집합 기준 성능 평가

ML1M_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4822 ±.0086		.5233 ±.0078		.6389 ±.0087		.7656 ±.0053	
선행	분할	.4638 ±.0042	(−3.8%)	.5071 ±.0034	(−3.1%)	.6230 ±.0063	(−2.5%)	.7561 ±.0056	(−1.2%)
	슬라이딩	.4805 ±.0048	(−0.4%)	.5205 ±.0041	(−0.5%)	.6425 ±.0055	(+0.6%)	.7656 ±.0021	(−)
제안	노이즈	.4889 ±.0039	(+1.4%)	.5287 ±.0052	(+1.0%)	.6504 ±.0041	(+1.8%)	.7728 ±.0019	(+0.9%)
	중복성	.4878 ±.0039	(+1.2%)	.5278 ±.0027	(+0.9%)	.6492 ±.0032	(+1.6%)	.7717 ±.0040	(+0.8%)
	마스킹	.4810 ±.0015	(−0.2%)	.5208 ±.0008	(−0.6%)	.6421 ±.0051	(+0.5%)	.7648 ±.0029	(−0.1%)
	유사대체	.4873 ±.0019	(+1.1%)	.5276 ±.0027	(+0.8%)	.6495 ±.0136	(+1.7%)	.7737 ±.0048	(+1.1%)

표 9-5. 무비렌즈 50% 부분집합 기준 성능 평가

ML1M_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.5225 ±.0098		.5592 ±.0091		.6796 ±.0057		.7929 ±.0044	
선행	분할	.5068 ±.0071	(−3.0%)	.5443 ±.0059	(−2.7%)	.6670 ±.0061	(−1.9%)	.7823 ±.0032	(−1.3%)
	슬라이딩	.5167 ±.0041	(−1.1%)	.5543 ±.0035	(−0.9%)	.6763 ±.0048	(−0.5%)	.7919 ±.0026	(−0.1%)
제안	노이즈	.5228 ±.0059	(+0.1%)	.5611 ±.0019	(+0.3%)	.6868 ±.0019	(+1.1%)	.7977 ±.0012	(+0.6%)
	중복성	.5270 ±.0025	(+0.9%)	.5631±.0021	(+0.7%)	.6877±.0026	(+1.2%)	.7989±.0018	(+0.8%)
	마스킹	.5198 ±.0038	(−0.5%)	.5558 ±.0033	(−0.6%)	.6797 ±.0047	(−)	.7906 ±.0030	(−0.3%)
	유사대체	.5259 ±.0031	(+0.7%)	.5622 ±.0019	(+0.5%)	.6852 ±.0043	(+0.8%)	.7967 ±.0011	(+0.5%)

표 9-6. 무비렌즈 전체 데이터셋 기준 성능 평가

무비렌즈(ML-1M) 데이터셋의 경우, 전체 데이터셋의 10~40% 를 사용한 경우 모든 증강 기법에서 유의미한 성능 향상이 확인되었다. 특히 전체의 10% 만을 사용한 데이터셋의 경우 제안된 노이즈 추가 기법을 적용하였을 경우 베이스라인 대비 NDCG@10 기준 약 28.5% 의 높은 성능 향상을 확인할 수 있었다.

다만 전체 데이터셋의 50% 이상 사용한 경우, 데이터 증강 기법 전반적으로 성능 향상 폭이 1% 내외로 감소하며 베이스라인과 크게 다르지 않은 성능을 나타냈다. 이를 통해 해당 데이터셋의 경우 50% 이상 사용하는 경우 데이터 증강으로 인한 개선 효과는 크게 기대하기 힘든 것을 확인할 수 있었다.

고왈라(Gowalla) 데이터셋 실험 결과

고왈라(Gowalla) 데이터셋에 대한 성능 평가 결과는 아래의 [표 10-1~10-6] 를 통해 확인할 수 있다.

Gowalla_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4864 ±.0027		.4981 ±.0022		.5163 ±.0032		.5524 ±.0017	
선행	**분할**	.4904 ±.0040	(+0.8%)	.5031 ±.0033	(+1.0%)	.5247 ±.0032	(+1.6%)	.5643 ±.0026	(+2.2%)
	슬라이딩	.4935 ±.0016	(+1.4%)	.5060 ±.0018	(+1.6%)	.5275 ±.0017	(+2.2%)	.5667 ±.0025	(+2.6%)
제안	**노이즈**	.5030 ±.0035	(+3.4%)	.5149 ±.0028	(+3.4%)	.5350 ±.0043	(+3.6%)	.5720 ±.0026	(+3.6%)
	중복성	.5025 ±.0014	(+3.3%)	.5147 ±.0014	(+3.3%)	.5341 ±.0026	(+3.5%)	.5719 ±.0025	(+3.5%)
	마스킹	.4965 ±.0008	(+2.1%)	.5088 ±.0006	(+2.2%)	.5286 ±.0018	(+2.4%)	.5667 ±.0011	(+2.6%)
	유사대체	.4986 ±.0048	(+2.5%)	.5115 ±.0047	(+2.7%)	.5312 ±.0055	(+2.9%)	.5711 ±.0049	(+3.4%)

표 10-1. 고왈라 10% 부분집합 기준 성능 평가

Gowalla_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.5465 ±.0016		.5612 ±.0017		.5856 ±.0028		.6314 ±.0029	
선행	**분할**	.5521 ±.0037	(+1.0%)	.5676 ±.0031	(+1.1%)	.5948 ±.0027	(+1.6%)	.6428 ±.0014	(+1.8%)
	슬라이딩	.5567 ±.0043	(+1.9%)	.5718 ±.0042	(+1.9%)	.5997 ±.0044	(+2.4%)	.6465 ±.0047	(+2.4%)
제안	**노이즈**	.5676 ±.0034	(+3.9%)	.5821 ±.0030	(+3.7%)	.6103 ±.0032	(+4.2%)	.6555 ±.0029	(+3.8%)
	중복성	.5693 ±.0034	(+4.2%)	.5841 ±.0030	(+4.1%)	.6117 ±.0048	(+4.5%)	.6575 ±.0041	(+4.2%)
	마스킹	.5605 ±.0027	(+2.6%)	.5754 ±.0026	(+2.5%)	.6027 ±.0020	(+2.9%)	.6489 ±.0047	(+2.8%)
	유사대체	.5584 ±.0027	(+2.2%)	.5736 ±.0025	(+2.2%)	.6010 ±.0028	(+2.6%)	.6479 ±.0028	(+2.6%)

표 10-2. 고왈라 20% 부분집합 기준 성능 평가

Gowalla_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6024 ±.0049		.6174 ±.0045		.6493 ±.0049		.6954 ±.0038	
선행	분할	.6112 ±.0092	(+1.5%)	.6269 ±.0082	(+1.5%)	.6582 ±.0088	(+1.4%)	.7067 ±.0058	(+1.6%)
	슬라이딩	.6149 ±.0062	(+2.1%)	.6304 ±.0062	(+2.1%)	.6633 ±.0076	(+2.2%)	.7112 ±.0075	(+2.3%)
제안	노이즈	.6177 ±.0020	(+2.5%)	.6334 ±.0026	(+2.6%)	.6669 ±.0019	(+2.7%)	.7156 ±.0050	(+2.9%)
	중복성	.6168 ±.0025	(+2.4%)	.6330 ±.0030	(+2.5%)	.6642 ±.0023	(+2.3%)	.7145 ±.0036	(+2.7%)
	마스킹	.6087 ±.0028	(+1.0%)	.6243 ±.0025	(+1.1%)	.6556 ±.0016	(+1.0%)	.7041 ±.0010	(+1.2%)
	유사대체	.6113 ±.0030	(+1.5%)	.6266 ±.0031	(+1.5%)	.6608 ±.0027	(+1.8%)	.7083 ±.0033	(+1.8%)

표 10-3. 고왈라 30% 부분집합 기준 성능 평가

Gowalla_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6419 ±.0050		.6579 ±.0046		.6922 ±.0048		.7417 ±.0039	
선행	분할	.6534 ±.0118	(+1.8%)	.6687 ±.0113	(+1.6%)	.7060 ±.0122	(+2.0%)	.7533 ±.0104	(+1.6%)
	슬라이딩	.6568 ±.0063	(+2.3%)	.6723 ±.0060	(+2.2%)	.7109 ±.0069	(+2.7%)	.7589 ±.0060	(+2.3%)
제안	노이즈	.6555 ±.0032	(+2.1%)	.6714 ±.0024	(+2.1%)	.7088 ±.0035	(+2.4%)	.7581 ±.0015	(+2.2%)
	중복성	.6563 ±.0022	(+2.2%)	.6716 ±.0019	(+2.1%)	.7079 ±.0021	(+2.3%)	.7550 ±.0017	(+1.8%)
	마스킹	.6499 ±.0020	(+1.2%)	.6658 ±.0017	(+1.2%)	.7005 ±.0022	(+1.2%)	.7497 ±.0013	(+1.1%)
	유사대체	.6511 ±.0021	(+1.4%)	.6663 ±.0026	(+1.3%)	.7037 ±.0032	(+1.7%)	.7508 ±.0045	(+1.2%)

표 10-4. 고왈라 40% 부분집합 기준 성능 평가

Gowalla_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6795 ±.0072		.6947 ±.0068		.7328 ±.0081		.7801 ±.0073	
선행	분할	.6915 ±.0128	(+1.8%)	.7064 ±.0121	(+1.7%)	.7457 ±.0144	(+1.8%)	.7917 ±.0118	(+1.5%)
	슬라이딩	.7006 ±.0088	(+3.1%)	.7152 ±.0087	(+2.9%)	.7578 ±.0096	(+3.4%)	.8029 ±.0092	(+2.9%)
제안	노이즈	.6922 ±.0020	(+1.9%)	.7074 ±.0020	(+1.8%)	.7495 ±.0025	(+2.3%)	.7967 ±.0021	(+2.1%)
	중복성	.6895 ±.0052	(+1.5%)	.7050 ±.0048	(+1.5%)	.7457 ±.0055	(+1.8%)	.7937 ±.0045	(+1.8%)
	마스킹	.6836 ±.0034	(+0.6%)	.6992 ±.0038	(+0.6%)	.7380 ±.0025	(+0.7%)	.7861 ±.0039	(+0.8%)
	유사대체	.6857 ±.0035	(+0.9%)	.7015 ±.0034	(+1.0%)	.7407 ±.0033	(+1.1%)	.7894 ±.0029	(+1.2%)

표 10-5. 고왈라 50% 부분집합 기준 성능 평가

Gowalla_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.7993 ±.0030		.8110 ±.0012		.8586 ±.0031		.8968 ±.0014	
선행	분할	.7980 ±.0031	(−0.2%)	.8109 ±.0032	(−)	.8595 ±.0037	(+0.1%)	.8990 ±.0041	(+0.2%)
	슬라이딩	.8169 ±.0055	(+2.2%)	.8279 ±.0053	(+2.1%)	.8802 ±.0062	(+2.5%)	.9139 ±.0056	(+1.9%)
제안	노이즈	.8007 ±.0036	(+0.2%)	.8134 ±.0034	(+0.3%)	.8636 ±.0050	(+0.6%)	.9028 ±.0045	(+0.7%)
	중복성	.7970 ±.0038	(−0.3%)	.8096 ±.0035	(−0.2%)	.8590 ±.0028	(−)	.8978 ±.0023	(+0.1%)
	마스킹	.7962 ±.0033	(−0.4%)	.8091 ±.0030	(−0.2%)	.8571 ±.0024	(−0.2%)	.8979 ±.0036	(+0.1%)
	유사대체	(Substitution 기법은 계산 비용이 지나치게 커져 실험 진행하지 않음).							

표 10-6. 고왈라 전체 데이터셋 기준 성능 평가

고왈라(Gowalla) 데이터셋의 경우, 전체의 10~20% 를 사용한 경우 제안하는 모든 증강 기법에서 2% 이상의 유의미한 성능 향상이 확인되었으나, 부분집합 크기가 증가함에 따라 성능 향상 폭이 감소하는 양상이 확인되었다. 비교적 규모가 큰 해당 데이터셋 특성상, 전체의 10% 만 사용하더라도 상대적으로 안정적인 성능을 보이며 데이터 증강에 따른 향상 폭도 드라마틱하게 크지는 않았다.

전체 데이터셋의 30~40%만 사용하여 학습을 진행한 경우에도 전반적으로 1~3% 수준의 성능 향상이 확인되며, 아이템 마스킹 방식을 제외한 기법들은 모두 유의미한 수준의 향상을 나타냈다. 다만 전체의 50% 이상 사용한 경우, 전반적으로 성능 향상 폭이 1% 내외로 감소하며 제안 기법 중에는 노이즈 추가 방식만이 유의한 성능 개선을 나타냈으며, 전체 데이터셋을 사용한 경우 베이스라인 대비 성능이 저하되는 경우도 일부 나타났다. 또한 40% 이상 사용한 경우 슬라이딩 윈도우 기법이 제안 기법 대비 근소하게 우수한 성능을 보이며 가장 적합한 기법으로 나타났다.

아마존 비디오게임(Games) 데이터셋 실험 결과

아마존 비디오게임(Games) 데이터셋에 대한 성능 평가 결과는 아래의 [표 11-1~11-6] 를 통해 확인할 수 있다.

Games_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.1504 ±.0055		.1830 ±.0059		.2170 ±.0074		.3184 ±.0096	
선행	**분할**	.1580 ±.0039	(+5.0%)	.1863 ±.0052	(+1.8%)	.2197 ±.0042	(+1.2%)	.3077 ±.0085	(−3.4%)
	슬라이딩	.1681 ±.0041	(+11.8%)	.1991 ±.0031	(+8.8%)	.2333 ±.0041	(+7.5%)	.3298 ±.0029	(+3.6%)
제안	**노이즈**	.2060 ±.0033	(+37.0%)	.2350 ±.0036	(+28.4%)	.2744 ±.0027	(+26.4%)	.3645 ±.0036	(+14.5%)
	중복성	.1949 ±.0053	(+29.0%)	.2252 ±.0045	(+23.1%)	.2647 ±.0049	(+22.0%)	.3609 ±.0031	(+13.4%)
	마스킹	.1713 ±.0113	(+13.9%)	.2022 ±.0099	(+10.5%)	.2377 ±.0095	(+9.5%)	.3336 ±.0076	(+4.8%)
	유사대체	.1566 ±.0049	(+4.1%)	.1887 ±.0047	(+3.1%)	.2260 ±.0060	(+4.1%)	.3256 ±.0069	(+2.2%)

표 11-1. 아마존 비디오게임 10% 부분집합 기준 성능 평가

Games_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.2103 ±.0085		.2462 ±.0088		.2961 ±.0106		.4072 ±.0120	
선행	**분할**	.2239 ±.0085	(+6.4%)	.2613 ±.0089	(+6.1%)	.3113 ±.0095	(+5.1%)	.4274 ±.0110	(+5.0%)
	슬라이딩	.2617 ±.0046	(+24.4%)	.2965 ±.0046	(+20.5%)	.3519 ±.0064	(+18.8%)	.4599 ±.0062	(+12.9%)
제안	**노이즈**	.2903 ±.0033	(+38.0%)	.3240 ±.0028	(+31.6%)	.3823 ±.0047	(+29.1%)	.4868 ±.0050	(+19.5%)
	중복성	.2874 ±.0015	(+36.6%)	.3217 ±.0008	(+30.7%)	.3796 ±.0030	(+28.2%)	.4860 ±.0038	(+19.4%)
	마스킹	.2734 ±.0024	(+30.0%)	.3081 ±.0024	(+25.2%)	.3638 ±.0019	(+22.8%)	.4713 ±.0041	(+15.8%)
	유사대체	.2667 ±.0043	(+26.8%)	.3015 ±.0043	(+22.5%)	.3570 ±.0032	(+20.5%)	.4647 ±.0028	(+14.1%)

표 11-2. 아마존 비디오게임 20% 부분집합 기준 성능 평가

Games_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3379 ±.0023		.3728 ±.0016		.4419 ±.0026		.5499 ±.0021	
선행	분할	.3016 ±.0037	(−10.7%)	.3393 ±.0032	(−9.0%)	.4054 ±.0044	(−8.3%)	.5218 ±.0031	(−3.4%)
	슬라이딩	.3161 ±.0048	(−10.7%)	.3532 ±.0047	(−9.0%)	.4225 ±.0096	(−8.3%)	.5374 ±.0096	(−3.4%)
제안	노이즈	.3417 ±.0003	(+1.1%)	.3774 ±.0012	(+1.2%)	.4457 ±.0031	(+0.8%)	.5558 ±.0027	(+1.1%)
	중복성	.3456 ±.0040	(+2.3%)	.3808 ±.0042	(+2.1%)	.4520 ±.0039	(+2.3%)	.5609 ±.0041	(+2.0%)
	마스킹	.3352 ±.0041	(−0.8%)	.3725 ±.0040	(−0.1%)	.4423 ±.0043	(+0.1%)	.5576 ±.0040	(+1.4%)
	유사대체	.3257 ±.0028	(−3.6%)	.3614 ±.0027	(−3.1%)	.4279 ±.0056	(−3.2%)	.5383 ±.0058	(−2.1%)

표 11-3. 아마존 비디오게임 30% 부분집합 기준 성능 평가

Games_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3638 ±.0023		.3993 ±.0017		.4729 ±.0042		.5834 ±.0023	
선행	분할	.3245 ±.0026	(−10.8%)	.3628 ±.0019	(−9.2%)	.4338 ±.0024	(−8.3%)	.5522 ±.0022	(−5.4%)
	슬라이딩	.3398 ±.0088	(−6.6%)	.3778 ±.0082	(−5.4%)	.4508 ±.0119	(−4.7%)	.5685 ±.0107	(−2.6%)
제안	노이즈	.3665 ±.0028	(+0.7%)	.4026 ±.0019	(+0.8%)	.4777 ±.0030	(+1.0%)	.5895 ±.0040	(+1.0%)
	중복성	.3627 ±.0009	(−0.3%)	.3990 ±.0012	(−0.1%)	.4741 ±.0014	(+0.3%)	.5862 ±.0006	(+0.5%)
	마스킹	.3546 ±.0029	(−2.5%)	.3918 ±.0025	(−1.9%)	.4630 ±.0039	(−2.1%)	.5780 ±.0028	(−0.9%)
	유사대체	.3491 ±.0031	(−4.0%)	.3860 ±.0030	(−3.3%)	.4599 ±.0039	(−2.7%)	.5741 ±.0036	(−1.6%)

표 11-4. 아마존 비디오게임 40% 부분집합 기준 성능 평가

Games_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3922 ±.0044		.4290 ±.0033		.5062 ±.0048		.6197 ±.0034	
선행	분할	.3471 ±.0043	(−11.5%)	.3855 ±.0031	(−10.1%)	.4616 ±.0043	(−8.8%)	.5805 ±.0021	(−6.3%)
	슬라이딩	.3589 ±.0065	(−8.5%)	.3976 ±.0071	(−7.3%)	.4774 ±.0082	(−5.7%)	.5970 ±.0098	(−3.7%)
제안	노이즈	.3931 ±.0016	(+0.2%)	.4293 ±.0018	(+0.1%)	.5095 ±.0022	(+0.7%)	.6212 ±.0030	(+0.2%)
	중복성	.3856 ±.0025	(−1.7%)	.4218 ±.0026	(−1.7%)	.5026 ±.0039	(−0.7%)	.6145 ±.0048	(−0.8%)
	마스킹	.3775 ±.0024	(−3.8%)	.4145 ±.0022	(−3.4%)	.4928 ±.0029	(−2.7%)	.6071 ±.0024	(−2.0%)
	유사대체	.3724 ±.0019	(−5.1%)	.4093 ±.0023	(−4.6%)	.4889 ±.0026	(−3.4%)	.6029 ±.0028	(−2.7%)

표 11-5. 아마존 비디오게임 50% 부분집합 기준 성능 평가

Games_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4684 ±.0051		.5017 ±.0044		.5941 ±.0034		.6968 ±.0031	
선행	분할	.4135 ±.0031	(−11.7%)	.4500 ±.0029	(−10.3%)	.5372 ±.0050	(−9.6%)	.6493 ±.0051	(−6.8%)
	슬라이딩	.4225 ±.0023	(−9.8%)	.4586 ±.0018	(−8.6%)	.5468 ±.0035	(−8.0%)	.6582 ±.0042	(−5.5%)
제안	노이즈	.4688 ±.0034	(+0.1%)	.5031 ±.0031	(+0.3%)	.5936 ±.0045	(−0.1%)	.6993 ±.0059	(+0.4%)
	중복성	.4604 ±.0037	(−1.7%)	.4938 ±.0030	(−1.6%)	.5817 ±.0022	(−2.1%)	.6848 ±.0021	(−1.7%)
	마스킹	.4488 ±.0032	(−4.2%)	.4830 ±.0038	(−3.7%)	.5707 ±.0036	(−3.9%)	.6763 ±.0053	(−2.9%)
	유사대체	.4508 ±.0028	(−3.7%)	.4853 ±.0025	(−3.3%)	.5766 ±.0041	(−2.9%)	.6832 ±.0031	(−2.0%)

표 11-6. 아마존 비디오게임 전체 데이터셋 기준 성능 평가

아마존 비디오게임(Games) 데이터셋의 경우 전체의 10~20% 를 사용한 경우 모든 증강 기법에서 베이스라인 대비 뚜렷한 성능 향상이 확인되었으며, 특히 10%만 사용한 경우 데이터 증강 방식에 따라 성능 향상의 폭에 있어 큰 차이가 나타났다.

한편, 전체의 30% 이상 사용한 경우 데이터 증강 기법에 따라 성능 향상 또는 하락 여부가 다르게 나타나는 모습이 특징적이다. 이때, 선행 기법들은 10% 내외 큰 폭의 성능 하락이 나타난 반면, 본 연구에서 제안한 기법들은 1-5% 내외로 상대적으로 하락 폭이 작았으며, 특히 노이즈 추가 기법은 원본 데이터셋 전체를 사용한 경우에도 베이스라인과 유사한 성능 수준을 유지하며 성능을 해치지 않는 것으로 나타났다.

데이터 증강을 통한 성능 향상 여부

본 연구에서 제안하는 데이터 증강 기법들을 적용할 경우, 대부분의 부분집합 데이터셋에서 성능 향상이 나타나는 것을 확인할 수 있었다. 제안하는 증강 기법이 적용된 전체 71 개 실험 케이스[5] 가운데 약 78.9% 에 해당하는 56 개 케이스에서 성능 향상이 확인되었으며, 특히 노이즈 추가 기법의 경우 모든 부분 집합에 대해 성능 향상이 나타나는 것을 관찰할 수 있었다.

또한, 데이터셋의 크기가 상대적으로 작을 때에 보다 큰 폭의 성능 개선이 나타나는 경향성이 존재하였으며, 반대로 데이터셋이 충분히 큰 경우엔 증강을 통해 추가적으로 생성된 데이터가 성능 개선에 도움이 되지 않고 오히려 노이즈로 작용하여 추천 성능을 저하시키는 경우도 나타났다.

구체적으로, 각 데이터셋에 대해 각 증강 기법을 적용했을 때 베이스라인 대비 NDCG@10 기준 성능 증감율을 아래의 [표 12-1~12-3] 와 [그래프 1-1~1-3] 를 통해 확인할 수 있으며, 이 때 전체적인 결과 비교를 위해 본 연구 제안 기법뿐 아니라 선행 제안 기법 두 가지도 모두 포함하여 정리하였다.

---

[5] 유사 아이템 대체 기법의 경우, 고왈라(Gowalla) 전체 데이터셋에 대해 적용 시 유사 아이템을 구하는 과정에서의 계산 비용이 지나치게 높아짐에 따라 해당 케이스는 실험 진행하지 않음.
따라서 [(3 개 데이터셋) x (6 개 부분집합) x (4 개 증강 기법) − 1] = 71 개 케이스

데이터셋	무비렌즈(ML-1M) (N=6,040)					
	분할	슬라이딩	노이즈	중복성	마스킹	유사대체
10%	21.6%	26.4%	25.3%	**28.5%**	27.0%	22.7%
20%	3.0%	6.3%	5.4%	**7.1%**	4.8%	5.3%
30%	-1.5%	0.5%	**2.8%**	2.0%	1.2%	1.6%
40%	-1.5%	1.3%	1.8%	**2.5%**	1.5%	2.0%
50%	-3.1%	-0.5%	**1.0%**	0.8%	-0.5%	0.8%
100%	-2.7%	-0.9%	0.3%	**0.7%**	-0.6%	0.5%

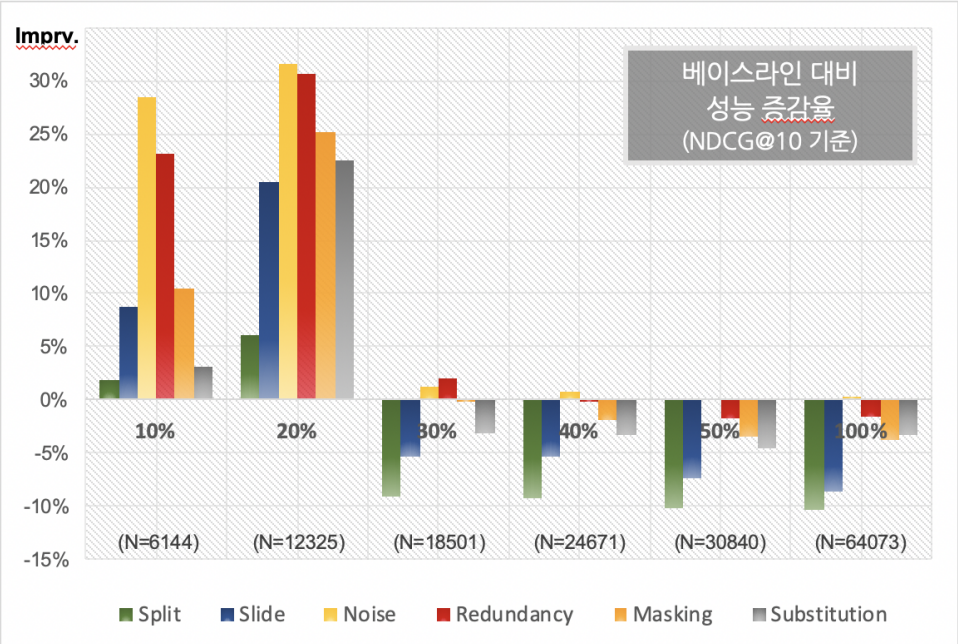
표 12-1. 무비렌즈 기준 각 증강 기법 적용 시 성능 증감율

데이터셋	고왈라(Gowalla) (N=85,034)					
	분할	슬라이딩	노이즈	중복성	마스킹	유사대체
10%	1.0%	1.6%	**3.4%**	3.3%	2.1%	2.7%
20%	1.1%	1.9%	3.7%	**4.1%**	2.5%	2.2%
30%	1.5%	2.1%	**2.6%**	2.5%	1.1%	1.5%
40%	1.6%	**2.2%**	2.1%	2.1%	1.2%	1.3%
50%	1.7%	**2.9%**	1.8%	1.5%	0.6%	1.0%
100%	0.0%	**2.1%**	0.3%	-0.2%	-0.2%	-

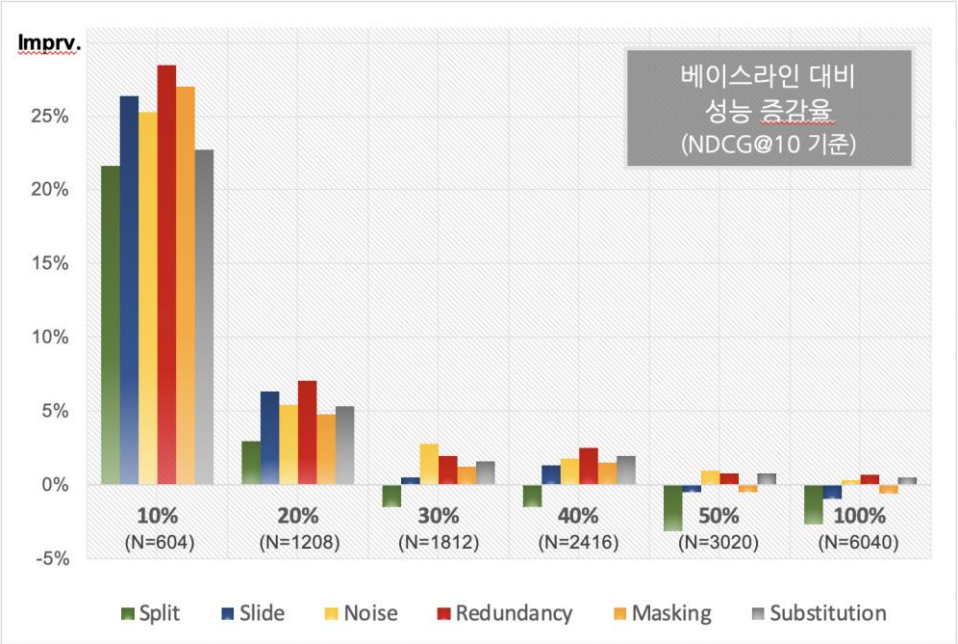
표 12-2. 고왈라 기준 각 증강 기법 적용 시 성능 증감율

데이터셋	아마존 비디오게임(Games) (N=64,073)					
	분할	슬라이딩	노이즈	중복성	마스킹	유사대체
10%	1.8%	8.8%	**28.4%**	23.1%	10.5%	3.1%
20%	6.1%	20.5%	**31.6%**	30.7%	25.2%	22.5%
30%	-9.0%	-5.3%	1.2%	**2.1%**	-0.1%	-3.1%
40%	-9.2%	-5.4%	**0.8%**	-0.1%	-1.9%	-3.3%
50%	-10.1%	-7.3%	**0.1%**	-1.7%	-3.4%	-4.6%
100%	-10.3%	-8.6%	**0.3%**	-1.6%	-3.7%	-3.3%

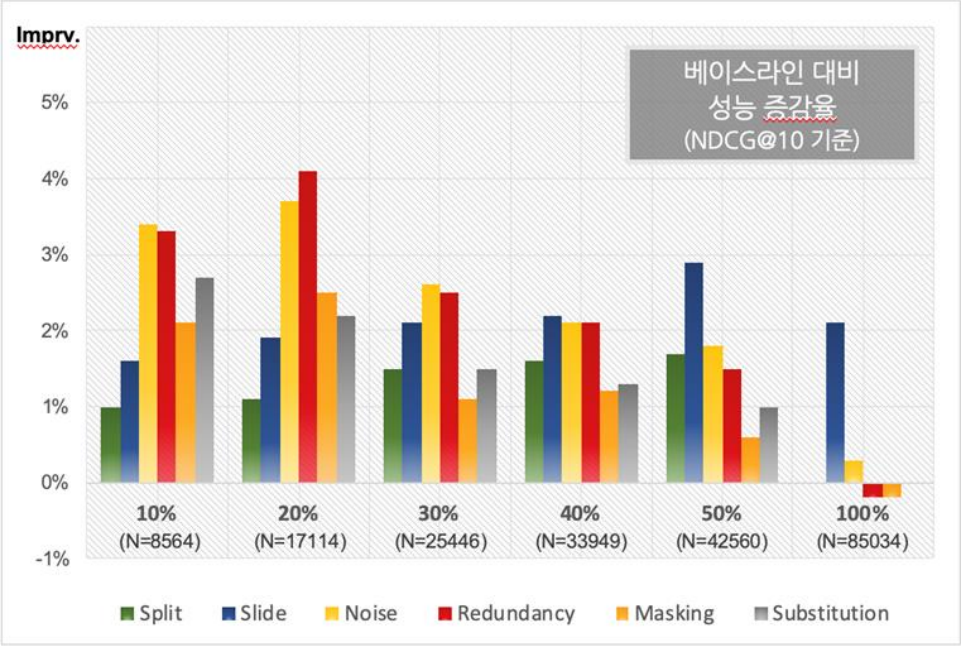
표 12-3. 아마존 비디오게임 기준 각 증강 기법 적용 시 성능 증감율



그래프 1-1. 무비렌즈 기준 각 증강 기법 적용 시 성능 증감율



그래프 1-2. 고왈라 기준 각 증강 기법 적용 시 성능 증감율



그래프 1-3. 아마존 비디오게임 기준 각 증강 기법 적용 시 성능 증감율

데이터 증강 방식에 따른 성능 변화 양상 차이

실험 결과, 동일한 데이터셋이라 할 지라도 적용되는 증강 기법에 따라 성능 변화가 다르게 나타났으나, 전반적으로 추가 (Injection) 계열 기법 및 슬라이딩 윈도우가 가장 효과적인 것으로 확인되었다.

아래의 [표 13] 은 각 부분집합 데이터셋 별 데이터 증강 규모 n_{aug} =10 을 기준으로 가장 높은 성능(best performance) 을 가져오는 최적 증강 기법과 해당 경우의 NDCG@10 기준의 성능 증가율을 정리한 결과이다. 최적 기법은 선행 기법과 본 연구 제안 기법을 구분하여 각각 표기하였으며, 선행 기법과 제안 기법을 아울러 해당 케이스의 최고 성능을 나타낸 경우에 대해 굵게(bold) 처리하여 강조하였다.

데이터셋	무비렌즈		고왈라		아마존 비디오게임	
	선행	제안	선행	제안	선행	제안
10%	슬라이딩 윈도우 (+26.4%)	**중복성 추가 (+28.5%)**	슬라이딩 윈도우 (+1.6%)	**노이즈 추가 (+3.4%)**	슬라이딩 윈도우 (+8.8%)	**노이즈 추가 (+28.4%)**
20%	슬라이딩 윈도우 (+6.3%)	**중복성 추가 (+7.1%)**	슬라이딩 윈도우 (+1.9%)	**중복성 추가 (+4.1%)**	슬라이딩 윈도우 (+20.5%)	**노이즈 추가 (+31.6%)**
30%	슬라이딩 윈도우 (+0.5%)	**노이즈 추가 (+2.8%)**	슬라이딩 윈도우 (+2.1%)	**노이즈 추가 (+2.6%)**	슬라이딩 윈도우 (−5.3%)	**중복성 추가 (+2.1%)**
40%	슬라이딩 윈도우 (+1.3%)	**중복성 추가 (+2.5%)**	**슬라이딩 윈도우 (+2.2%)**	중복성 추가 (+2.1%)	슬라이딩 윈도우 (−5.4%)	**노이즈 추가 (+0.8%)**
50%	슬라이딩 윈도우 (−0.5%)	**노이즈 추가 (+1.0%)**	**슬라이딩 윈도우 (+2.9%)**	노이즈 추가 (+1.8%)	슬라이딩 윈도우 (−7.3%)	**노이즈 추가 (+0.1%)**
100%	슬라이딩 윈도우 (−0.9%)	**중복성 추가 (+0.7%)**	**슬라이딩 윈도우 (+2.1%)**	노이즈 추가 (+0.3%)	슬라이딩 윈도우 (−8.6%)	**노이즈 추가 (+0.3%)**

표 13. 데이터셋 별 최적 증강 기법 적용 시 성능 증가율

전체 18개 케이스 가운데 절반인 9개 부분집합 데이터셋에 대해서 본 연구에서 제안한 노이즈 추가 기법이 가장 큰 성능 향상을 가져오는 것으로 나타났으며, 이어서 중복성 추가(6개), 슬라이딩 윈도우(3개) 가 뒤를 이었다. 특히 노이즈/중복성 추가 기법은 상대적으로 데이터셋 규모가 작은 무비렌즈(ML-1M) 와 아마존 비디오게임(Games) 데이터셋의 10%만 사용한 경우 30% 에 가까운 큰 폭의 성능 향상을 가져오는 것이 주목할 만한 결과이다.

반면, 아이템 마스킹, 유사 아이템 대체 기법의 경우 이들 추가 계열 기법에 비해 상대적으로 더 작은 규모에서도 더 이상 성능이 개선되지 않는 상태, 곧 성능 포화 상태(saturation) 에 이르는 것으로 나타났으며, 때로는 단순히 성능 개선 효과가 나타나지 않는 것을 넘어 베이스라인 대비 성능을 크게 악화시키기도 하는 것으로 확인되었다. 이처럼 해당 기법들의 경우 데이터셋에 따라 데이터 증강이 오히려 성능을 하락시키는 결과를 낳을 수도 있으므로 신중한 적용이 필요할 것으로 보인다. 성능 포화에 관해서는 뒤의 논의 단계에서 보다 자세히 다룰 예정이다.

데이터 증강 규모에 따른 성능 변화 양상 차이

한편, 선행 연구에 따르면 데이터 증강 규모는 증강 효과에 중요한 영향을 미치는 핵심 하이퍼 파라미터로서 각 데이터셋에 적합한 값으로 튜닝하는 것이 필수적이다[5]. 본 연구에서는 기본적으로 모든 데이터셋에 대해 동일한 증강 규모 n_{aug} = 10 를 적용하여 실험을 진행하였으며, 해당 값에 따른 성능의 변화 양상은 선행 기법과 본 제안 기법 각각에서 가장 뛰어난 효과를 나타낸 슬라이딩 윈도우와 노이즈 추가 기법만을 대상으로 진행하였다. 증강 규모의 범위는 본 실험에서의 기준값인 n_{aug} = 10 을 포함하여 $n_{aug} \in \{2,5,10,15\}$ 로 설정하였다.

결과적으로 실험을 진행한 모든 데이터셋에 대해 성능 향상 폭은 데이터 증강 규모와 비례하는 경향이 나타났다. 또한 데이터셋 종류 및

증강 기법과 무관하게 데이터 증강 규모가 지나치게 작을 경우 데이터 증강에 의한 성능 향상을 기대하기 어려운 것을 확인할 수 있었다.

단적으로, 데이터 증강 규모가 n_{aug} = 2 인 경우 증강 기법과 데이터셋을 불문하고 베이스라인 대비 큰 폭의 성능 하락을 관찰할 수 있었다. 이는 곧 데이터 증강을 통해 학습 데이터의 규모가 증가하긴 하였으나 해당 데이터셋은 원본 데이터셋에 내재한 패턴, 곧 데이터셋의 특성을 해치는 결과를 가져온 것이라고 해석할 수 있을 것이다. 결과적으로 모델의 입장에서 무엇이 원본 데이터인지 구분할 수 없는 상황을 초래하여 왜곡된 패턴을 학습하게 됨에 따라 추천 성능이 하락하게 되는 것이다.

데이터 증강 규모가 기준 값의 절반 수준인 n_{aug} = 5 인 경우, 일부 데이터셋에 대해 전체의 10~20%만 사용하였을 때는 성능 개선 효과가 나타나기도 하였으나, 그 외에는 베이스라인과 유사한 수준의 성능을 나타내거나 오히려 성능 하락을 가져오기도 하였다.

한편 데이터 증강 규모를 n_{aug} = 15 로 기준 값보다 더 크게 설정하여 적용한 경우, 기준 값에서 성능이 향상되었던 경우는 더욱 큰 폭의 성능 향상이 나타났으며, 기준 값에서 성능이 향상되지 않거나 오히려 저하되었던 경우에도 성능이 향상되는 결과를 확인할 수 있었다. 노이즈 추가 기법의 경우, 모든 데이터셋에 대해서 n_{aug} = 10 일 때는 전체 데이터셋을 사용하였을 때 1% 미만의 근소한 성능 향상을 보이며 사실상 베이스라인과 큰 차이가 없었다. 하지만 증강 규모를 n_{aug} = 15 로 증가시키자 전체 데이터셋을 사용한 경우에도 유의한 수준의 향상이 나타나는 것이 확인되었다. 슬라이딩 윈도우 기법도 마찬가지로, 기준

값일 때는 무비렌즈 데이터셋의 50% 이상을 사용하였을 때 근소한 성능 하락이 나타났으나, n_{aug} = 15 으로 증가시킨 결과 베이스라인과 유사한 성능 수준을 회복하였다. 다만, 아마존 비디오게임과 같이 기준값에서의 성능 하락 폭이 5% 이상으로 크게 나타났을 경우 증강 규모를 늘리더라도 성능 하락을 피하지는 못하였다.

이러한 결과들을 종합해볼 때, 데이터 증강을 적용하는 데이터셋 규모가 큰 경우 성능 개선 효과를 기대하기 위해서는 더 큰 규모의 데이터 증강을 필요로 한다는 것을 추론할 수 있다. 각 데이터셋 별로 증강 규모에 따른 성능 변화 양상은 아래 [표 14~16] 및 [그래프 2-1~2-3] 를 통해 구체적으로 확인 가능하다.

ML1M_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.2518 ±.0156		.3114 ±.0118		.3705 ±.0149		.5493 ±.0125	
선행	$n_{aug}=2$	.1953 ±.0219	(−22.4%)	.2475 ±.0140	(−20.5%)	.3041 ±.0397	(−17.9%)	.4669 ±.0211	(−15.0%)
	$n_{aug}=5$	.2809 ±.0085	(+11.6%)	.3424 ±.0018	(+10.0%)	.4189 ±.0116	(+13.1%)	.6060 ±.0216	(+10.3%)
	$n_{aug}=10$	.3401 ±.0091	(+35.1%)	.3936 ±.0045	(+26.4%)	.4821 ±.0143	(+30.1%)	.6473 ±.0119	(+17.8%)
	$n_{aug}=15$	**.3649 ±.0166**	**(+44.9%)**	**.4104 ±.0132**	**(+31.8%)**	**.5121 ±.0124**	**(+38.2%)**	**.6540 ±.0029**	**(+19.1%)**
제안	$n_{aug}=2$	.1920 ±.0105	(−23.7%)	.2498 ±.0195	(−19.8%)	.2974 ±.0191	(−19.7%)	.4785 ±.0219	(−12.9%)
	$n_{aug}=5$	.2952 ±.0121	(+17.3%)	.3484 ±.0120	(+11.9%)	.4363 ±.0079	(+17.7%)	.6010 ±.0103	(+9.4%)
	$n_{aug}=10$	.3381 ±.0071	(+34.3%)	.3902 ±.0038	(+25.3%)	.4818 ±.0134	(+30.0%)	.6427 ±.0160	(+17.0%)
	$n_{aug}=15$	.3623 ±.0104	(+43.9%)	.4087 ±.0106	(+31.2%)	.5061 ±.0141	(+36.6%)	.6479 ±.0110	(+18.0%)

표 14-1. 무비렌즈 10% 부분집합 기준 증강 규모 별 성능 평가

ML1M_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3791 ±.0134		.4305 ±.0104		.5270 ±.0151		.6858 ±.0069	
선행	$n_{aug}=2$	.2992 ±.0133	(−21.1%)	.3543 ±.0116	(−17.7%)	.4404 ±.0186	(−16.4%)	.6106 ±.0164	(−11.0%)
	$n_{aug}=5$	.3741 ±.0032	(−1.3%)	.4232 ±.0028	(−1.7%)	.5287 ±.0050	(+0.3%)	.6805 ±.0065	(−0.8%)
	$n_{aug}=10$	.4125 ±.0065	(+8.8%)	.4576 ±.0038	(+6.3%)	.5659 ±.0152	(+7.4%)	.7050 ±.0089	(+2.8%)
	$n_{aug}=15$	**.4256 ±.0060**	**(+12.3%)**	**.4697 ±.0057**	**(+9.1%)**	.5748 ±.0043	(+9.1%)	.7108 ±.0047	(+3.7%)
제안	$n_{aug}=2$	.3229 ±.0100	(−14.8%)	.3796 ±.0070	(−11.8%)	.4674 ±.0219	(−11.3%)	.6416 ±.0169	(−6.4%)
	$n_{aug}=5$	.3799 ±.0031	(+0.2%)	.4337 ±.0048	(+0.7%)	.5298 ±.0044	(+0.5%)	.6956 ±.0089	(+1.4%)
	$n_{aug}=10$	.4088 ±.0044	(+7.8%)	.4536 ±.0042	(+5.4%)	.5639 ±.0033	(+7.0%)	.7025 ±.0.39	(+2.4%)
	$n_{aug}=15$	.4251 ±.0072	(+12.1%)	.4683 ±.0044	(+8.8%)	**.5825 ±.0132**	**(+10.5%)**	**.7158 ±.0047**	**(+4.4%)**

표 14-2. 무비렌즈 20% 부분집합 기준 증강 규모 별 성능 평가

ML1M_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4454 ±.0066		.4876 ±.0053		.6041 ±.0097		.7340 ±.0082	
선행	$n_{aug}=2$	.3549 ±.0088	(−20.3%)	.4098 ±.0056	(−16.0%)	.5088 ±.0163	(−15.8%)	.6778 ±.0059	(−7.6%)
	$n_{aug}=5$	.4261 ±.0027	(−4.3%)	.4712 ±.0033	(−3.4%)	.5780 ±.0036	(−4.3%)	.7174 ±.0045	(−2.3%)
	$n_{aug}=10$	.4469 ±.0071	(+0.3%)	.4903 ±.0057	(+0.5%)	.6000 ±.0091	(−0.7%)	.7336 ±.0041	(−0.1%)
	$n_{aug}=15$	**.4650 ±.0012**	**(+4.4%)**	**.5071 ±.0014**	**(+4.0%)**	**.6186 ±.0039**	**(+2.4%)**	**.7485 ±.0061**	**(+2.0%)**
제안	$n_{aug}=2$	.3785 ±.0066	(−15.0%)	.4299 ±.0060	(−11.8%)	.5341 ±.0032	(−11.6%)	.6926 ±.0040	(−5.6%)
	$n_{aug}=5$	.4298 ±.0021	(−3.5%)	.4762 ±.0034	(−2.4%)	.5845 ±.0061	(−3.2%)	.7278 ±.0065	(−0.8%)
	$n_{aug}=10$	.4576 ±.0029	(+2.8%)	.5012 ±.0035	(+2.8%)	.6116 ±.0025	(+1.3%)	.7462 ±.0066	(+1.7%)
	$n_{aug}=15$	.4622 ±.0085	(+3.8%)	.5044 ±.0061	(+3.4%)	.6179 ±.0070	(+2.3%)	.7465 ±.0022	(+1.7%)

표 14-3. 무비렌즈 30% 부분집합 기준 증강 규모 별 성능 평가

ML1M_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4671 ±.0085		.5090 ±.0086		.6276 ±.0081		.7566 ±.0078	
선행	$n_{aug}=2$	.4014 ±.0069	(−14.1%)	.4512 ±.0042	(−11.4%)	.5607 ±.0132	(−10.7%)	.7142 ±.0049	(−5.6%)
	$n_{aug}=5$	.4597 ±.0049	(−1.6%)	.5021 ±.0027	(−1.3%)	.6199 ±.0036	(−1.2%)	.7506 ±.0082	(−0.8%)
	$n_{aug}=10$	.4751 ±.0022	(+1.7%)	.5155 ±.0023	(+1.3%)	.6371 ±.0013	(+1.5%)	.7611 ±.0019	(+0.6%)
	$n_{aug}=15$	**.4829 ±.0027**	**(+3.4%)**	**.5239 ±.0026**	**(+2.9%)**	**.6412 ±.0022**	**(+2.2%)**	**.7668 ±.0024**	**(+1.3%)**
제안	$n_{aug}=2$	.4207 ±.0040	(−9.9%)	.4672 ±.0036	(−8.2%)	.5822 ±.0067	(−7.2%)	.7256 ±.0059	(−4.1%)
	$n_{aug}=5$	.4637 ±.0015	(−0.7%)	.5068 ±.0029	(−0.4%)	.6256 ±.0037	(−0.3%)	.7583 ±.0091	(+0.2%)
	$n_{aug}=10$	.4773 ±.0042	(+2.2%)	.5179 ±.0027	(+1.8%)	.6387 ±.0043	(+1.8%)	.7635 ±.0030	(+0.9%)
	$n_{aug}=15$	.4805 ±.0022	(+2.9%)	.5197 ±.0011	(+2.1%)	.6406 ±.0035	(+2.1%)	.7612 ±.0051	(+1.7%)

표 14-4. 무비렌즈 40% 부분집합 기준 증강 규모 별 성능 평가

ML1M_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4822 ±.0086		.5233 ±.0078		.6389 ±.0087		.7656 ±.0053	
선행	$n_{aug}=2$	.4258 ±.0034	(−11.7%)	.4703 ±.0027	(−10.1%)	.5889 ±.0007	(−7.8%)	.7265 ±.0018	(−5.1%)
	$n_{aug}=5$	.4692 ±.0027	(−2.7%)	.5096 ±.0015	(−2.6%)	.6311 ±.0051	(−1.2%)	.7555 ±.0031	(−1.3%)
	$n_{aug}=10$	.4805 ±.0048	(−0.4%)	.5205 ±.0041	(−0.5%)	.6425 ±.0055	(+0.6%)	.7656 ±.0021	(−)
	$n_{aug}=15$	**.4940 ±.0026**	**(+2.4%)**	**.5335 ±.0021**	**(+1.9%)**	**.6519 ±.0056**	**(+2.0%)**	**.7735 ±.0042**	**(+1.0%)**
제안	$n_{aug}=2$	.4355 ±.0008	(−9.7%)	.4806 ±.0015	(−8.2%)	.6002 ±.0045	(−6.1%)	.7386 ±.0016	(−3.5%)
	$n_{aug}=5$	.4754 ±.0053	(−1.4%)	.5175 ±.0044	(−1.1%)	.6358 ±.0049	(−0.5%)	.7653 ±.0042	(−)
	$n_{aug}=10$	.4889 ±.0039	(+1.4%)	.5287 ±.0052	(+1.0%)	.6504 ±.0041	(+1.8%)	.7728 ±.0019	(+0.9%)
	$n_{aug}=15$	.4924 ±.0042	(+2.1%)	.5315 ±.0038	(+1.6%)	.6516 ±.0078	(+2.0%)	.7739 ±.0049	(+1.1%)

표 14-5. 무비렌즈 50% 부분집합 기준 증강 규모 별 성능 평가

ML1M_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.5225 ±.0098		.5592 ±.0091		.6796 ±.0057		.7929 ±.0044	
선행	$n_{aug}=2$	.4929 ±.0051	(−5.7%)	.5332 ±.0011	(−4.7%)	.6528 ±.0061	(−3.9%)	.7769 ±.0055	(−2.0%)
	$n_{aug}=5$	.5163 ±.0030	(−1.2%)	.5537 ±.0020	(−1.0%)	.6742 ±.0027	(−0.8%)	.7895 ±.0021	(−0.4%)
	$n_{aug}=10$	.5167 ±.0041	(−1.1%)	.5543 ±.0035	(−0.9%)	.6763 ±.0048	(−0.5%)	.7919 ±.0026	(−0.1%)
	$n_{aug}=15$	.5276 ±.0025	(+1.0%)	.5635 ±.0014	(+0.8%)	.6842 ±.0032	(+0.7%)	.7945 ±.0026	(+0.2%)
제안	$n_{aug}=2$	.5039 ±.0014	(−3.6%)	.5417 ±.0016	(−3.1%)	.6642 ±.0018	(−2.3%)	.7806 ±.0028	(−1.6%)
	$n_{aug}=5$	.5216 ±.0014	(−0.2%)	.5574 ±.0010	(−0.3%)	.6821 ±.0016	(+0.4%)	.7922 ±.0018	(−0.1%)
	$n_{aug}=10$	.5228 ±.0059	(+0.1%)	.5611 ±.0019	(+0.3%)	.6868 ±.0019	(+1.1%)	.7977 ±.0012	(+0.6%)
	$n_{aug}=15$	**.5311 ±.0044**	**(+1.7%)**	**.5662 ±.0031**	**(+1.2%)**	**.6883 ±.0047**	**(+1.3%)**	**.7961 ±.0023**	**(+0.4%)**

표 14-6. 무비렌즈 전체 데이터셋 기준 증강 규모 별 성능 평가

Gowalla_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4864 ±.0027		.4981 ±.0022		.5163 ±.0032		.5524 ±.0017	
선행	$n_{aug}=2$	.4027 ±.0006	(−17.2%)	.4196 ±.0033	(−15.8%)	.4692 ±.0007	(−9.1%)	.5215 ±.0004	(−5.6%)
	$n_{aug}=5$	.4736 ±.0029	(−2.6%)	.4868 ±.0030	(−2.3%)	.5099 ±.0020	(−1.2%)	.5509 ±.0021	(−0.3%)
	$n_{aug}=10$	.4935 ±.0016	(+1.4%)	.5060 ±.0018	(+1.6%)	.5275 ±.0017	(+2.2%)	.5667 ±.0025	(+2.6%)
	$n_{aug}=15$	.5070 ±.0020	(+4.2%)	.5194 ±.0024	(+4.3%)	.5398 ±.0032	(+4.6%)	.5785 ±.0047	(+4.7%)
제안	$n_{aug}=2$	.4148 ±.0058	(−14.7%)	.4307 ±.0056	(−13.5%)	.4813 ±.0072	(−6.8%)	.5304 ±.0066	(−4.0%)
	$n_{aug}=5$	.4823 ±.0014	(−0.8%)	.4949 ±.0008	(−0.6%)	.5188 ±.0023	(+0.5%)	.5582 ±.0017	(+1.0%)
	$n_{aug}=10$	.5030 ±.0035	(+3.4%)	.5149 ±.0028	(+3.4%)	.5350 ±.0043	(+3.6%)	.5720 ±.0026	(+3.6%)
	$n_{aug}=15$	.5135 ±.0015	(+5.6%)	.5260 ±.0009	(+5.6%)	.5442 ±.0033	(+5.4%)	.5832 ±.0019	(+5.6%)

표 15-1. 고왈라 10% 부분집합 기준 증강 규모 별 성능 평가

Gowalla_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.5465 ±.0016		.5612 ±.0017		.5856 ±.0028		.6314 ±.0029	
선행	$n_{aug}=2$	.4635 ±.0006	(−15.2%)	.4804 ±.0008	(−14.4%)	.5251 ±.0023	(−10.3%)	.5777 ±.0042	(−8.5%)
	$n_{aug}=5$	.5248 ±.0071	(−4.0%)	.5406 ±.0069	(−3.7%)	.5485 ±.0079	(−2.9%)	.6175 ±.0075	(−2.2%)
	$n_{aug}=10$	.5567 ±.0043	(+1.9%)	.5718 ±.0042	(+1.9%)	.5997 ±.0044	(+2.4%)	.6465 ±.0047	(+2.4%)
	$n_{aug}=15$	.5812 ±.0029	(+6.4%)	.5962 ±.0029	(+6.2%)	.6248 ±.0033	(+6.7%)	.6716 ±.0036	(+6.4%)
제안	$n_{aug}=2$	.4739 ±.0069	(−13.3%)	.4918 ±.0064	(−12.4%)	.5327 ±.0074	(−9.0%)	.5879 ±.0059	(−6.9%)
	$n_{aug}=5$	.5400 ±.0018	(−1.2%)	.5548 ±.0018	(−1.1%)	.5845 ±.0004	(−0.2%)	.6307 ±.0017	(−0.1%)
	$n_{aug}=10$	.5676 ±.0034	(+3.9%)	.5821 ±.0030	(+3.7%)	.6103 ±.0032	(+4.2%)	.6555 ±.0029	(+3.8%)
	$n_{aug}=15$	.5825 ±.0011	(+6.6%)	.5973 ±.0021	(+6.4%)	.6241 ±.0028	(+6.6%)	.6701 ±.0057	(+6.1%)

표 15-2. 고왈라 20% 부분집합 기준 증강 규모 별 성능 평가

Gowalla_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6024 ±.0049		.6174 ±.0045		.6493 ±.0049		.6954 ±.0038	
선행	$n_{aug}=2$	.5048 ±.0006	(−16.2%)	.5232 ±.0004	(−15.3%)	.5637 ±.0006	(−13.2%)	.6213 ±.0008	(−10.7%)
	$n_{aug}=5$	.5724 ±.0045	(−5.0%)	.5889 ±.0047	(−4.6%)	.6208 ±.0041	(−4.4%)	.6719 ±.0058	(−3.4%)
	$n_{aug}=10$	.6149 ±.0062	(+2.1%)	.6304 ±.0062	(+2.1%)	.6633 ±.0076	(+2.2%)	.7112 ±.0075	(+2.3%)
	$n_{aug}=15$	.6369 ±.0028	(+5.7%)	.6521 ±.0030	(+5.6%)	.6865 ±.0026	(+5.7%)	.7335 ±.0032	(+5.5%)
제안	$n_{aug}=2$	.5172 ±.0031	(−14.1%)	.5352 ±.0028	(−13.3%)	.5755 ±.0017	(−11.4%)	.6313 ±.0003	(−9.2%)
	$n_{aug}=5$	.5845 ±.0021	(−3.0%)	.6008 ±.0026	(−2.7%)	.6333 ±.0014	(−2.5%)	.6848 ±.0017	(−1.5%)
	$n_{aug}=10$	.6177 ±.0020	(+2.5%)	.6334 ±.0026	(+2.6%)	.6669 ±.0019	(+2.7%)	.7156 ±.0050	(+2.9%)
	$n_{aug}=15$	**.6376 ±.0012**	**(+5.8%)**	**.6525 ±.0013**	**(+5.7%)**	**.6870 ±.0027**	**(+5.8%)**	**.7332 ±.0026**	**(+5.4%)**

표 15-3. 고왈라 30% 부분집합 기준 증강 규모 별 성능 평가

Gowalla_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6419 ±.0050		.6579 ±.0046		.6922 ±.0048		.7417 ±.0039	
선행	$n_{aug}=2$	.5264 ±.0118	(−18.0%)	.5454 ±.0113	(−17.1%)	.5855 ±.0122	(−15.4%)	.6443 ±.0104	(−13.1%)
	$n_{aug}=5$	.6072 ±.0018	(−5.4%)	.6243 ±.0020	(−5.1%)	.6597 ±.0027	(−4.7%)	.7125 ±.0037	(−3.9%)
	$n_{aug}=10$	.6568 ±.0063	(+2.3%)	.6723 ±.0060	(+2.2%)	.7109 ±.0069	(+2.7%)	.7589 ±.0060	(+2.3%)
	$n_{aug}=15$	**.6863 ±.0036**	**(+6.9%)**	**.7007 ±.0036**	**(+6.5%)**	**.7383 ±.0042**	**(+6.7%)**	**.7829 ±.0047**	**(+5.5%)**
제안	$n_{aug}=2$	.5447 ±.0032	(−15.1%)	.5624 ±.0024	(−14.5%)	.6052 ±.0035	(−12.6%)	.6600 ±.0015	(−11.0%)
	$n_{aug}=5$	.6133 ±.0097	(−4.5%)	.6305 ±.0096	(−4.2%)	.6669 ±.0100	(−3.6%)	.7202 ±.0094	(−2.9%)
	$n_{aug}=10$	.6499 ±.0020	(+1.2%)	.6658 ±.0017	(+1.2%)	.7005 ±.0022	(+1.2%)	.7497 ±.0013	(+1.1%)
	$n_{aug}=15$	.6781 ±.0031	(+5.6%)	.6928 ±.0032	(+5.3%)	.7307 ±.0026	(+5.6%)	.7760 ±.0030	(+4.6%)

표 15-4. 고왈라 40% 부분집합 기준 증강 규모 별 성능 평가

Gowalla_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.6795 ±.0072		.6947 ±.0068		.7328 ±.0081		.7801 ±.0073	
선행	$n_{aug}=2$	.5623 ±.0128	(−17.2%)	.5815 ±.0121	(−16.3%)	.6212 ±.0144	(−15.2%)	.6809 ±.0118	(−12.7%)
	$n_{aug}=5$	.6436 ±.0018	(−5.3%)	.6595 ±.0023	(−5.1%)	.7008 ±.0024	(−4.4%)	.7498 ±.0033	(−3.9%)
	$n_{aug}=10$	.7006 ±.0088	(+3.1%)	.7152 ±.0087	(+2.9%)	.7578 ±.0096	(+3.4%)	.8029 ±.0092	(+2.9%)
	$n_{aug}=15$	**.7245 ±.0023**	**(+6.6%)**	**.7385 ±.0020**	**(+6.3%)**	**.7819 ±.0028**	**(+6.7%)**	**.8252 ±.0022**	**(+5.8%)**
제안	$n_{aug}=2$	.5736 ±.0020	(−15.6%)	.5931 ±.0020	(−14.6%)	.6355 ±.0025	(−13.3%)	.6959 ±.0021	(−10.8%)
	$n_{aug}=5$	.6541 ±.0096	(−3.7%)	.6710 ±.0090	(−3.4%)	.7099 ±.0082	(−3.1%)	.7622 ±.0066	(−2.3%)
	$n_{aug}=10$	.6922 ±.0020	(+1.9%)	.7074 ±.0020	(+1.8%)	.7495 ±.0025	(+2.3%)	.7967 ±.0021	(+2.1%)
	$n_{aug}=15$	.7094 ±.0015	(+4.4%)	.7241 ±.0022	(+4.2%)	.7655 ±.0026	(+4.5%)	.8108 ±.0052	(+3.9%)

표 15-5. 고왈라 50% 부분집합 기준 증강 규모 별 성능 평가

Gowalla_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.7993 ±.0030		.8110 ±.0012		.8586 ±.0031		.8968 ±.0014	
선행	$n_{aug}=2$	.6906 ±.0031	(−13.6%)	.7102 ±.0032	(−12.4%)	.7616 ±.0037	(−11.3%)	.8221 ±.0041	(−8.3%)
	$n_{aug}=5$	.7733 ±.0033	(−3.3%)	.7874 ±.0034	(−2.9%)	.8398 ±.0032	(−2.2%)	.8832 ±.0038	(−1.5%)
	$n_{aug}=10$	.8169 ±.0055	(+2.2%)	.8279 ±.0053	(+2.1%)	.8802 ±.0062	(+2.5%)	.9139 ±.0056	(+1.9%)
	$n_{aug}=15$	**.8265 ±.0038**	**(+3.4%)**	**.8374 ±.0034**	**(+3.3%)**	**.8880 ±.0008**	**(+3.4%)**	**.9214 ±.0003**	**(+2.7%)**
제안	$n_{aug}=2$	.7020 ±.0036	(−12.2%)	.7198 ±.0034	(−11.2%)	.7713 ±.0050	(−10.2%)	.8264 ±.0045	(−7.9%)
	$n_{aug}=5$	.7786 ±.0069	(−2.6%)	.7921 ±.0064	(−2.3%)	.8463 ±.0058	(−1.4%)	.8877 ±.0050	(−1.0%)
	$n_{aug}=10$	.7962 ±.0033	(−0.4%)	.8091 ±.0030	(−0.2%)	.8571 ±.0024	(−0.2%)	.8979 ±.0036	(+0.1%)
	$n_{aug}=15$	.8135 ±.0020	(+1.8%)	.8249 ±.0012	(+1.7%)	.8735 ±.0040	(+1.7%)	.9089 ±.0021	(+1.4%)

표 15-6. 고왈라 전체 데이터셋 기준 증강 규모 별 성능 평가

Games_10%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.1504 ±.0055		.1830 ±.0059		.2170 ±.0074		.3184 ±.0096	
선행	$n_{aug}=2$	.1120 ±.0052	(−25.5%)	.1429 ±.0044	(−21.9%)	.1647 ±.0066	(−24.1%)	.2611 ±.0061	(−18.0%)
	$n_{aug}=5$	.1474 ±.0034	(−2.0%)	.1790 ±.0036	(−2.2%)	.2099 ±.0036	(−3.3%)	.3085 ±.0043	(−3.1%)
	$n_{aug}=10$	.1681 ±.0041	(+11.8%)	.1991 ±.0031	(+8.8%)	.2333 ±.0041	(+7.5%)	.3298 ±.0029	(+3.6%)
	$n_{aug}=15$	.1781 ±.0011	(+18.4%)	.2076 ±.0023	(+13.5%)	.2467 ±.0020	(+13.7%)	.3383 ±.0055	(+6.3%)
제안	$n_{aug}=2$	.1216 ±.0053	(−19.2%)	.1532 ±.0053	(−16.4%)	.1779 ±.0066	(−18.0%)	.2766 ±.0064	(−13.1%)
	$n_{aug}=5$	.1761 ±.0052	(+17.1%)	.2069 ±.0058	(+13.1%)	.2404 ±.0066	(+10.8%)	.3361 ±.0085	(+5.5%)
	$n_{aug}=10$	.2060 ±.0033	(+37.0%)	.2350 ±.0036	(+28.4%)	.2744 ±.0027	(+26.4%)	.3645 ±.0036	(+14.5%)
	$n_{aug}=15$	.2158 ±.0035	(+43.5%)	.2453 ±.0028	(+34.6%)	.2863 ±.0065	(+31.9%)	.3778 ±.0041	(+18.6%)

표 16-1. 아마존 비디오게임 10% 부분집합 기준 증강 규모 별 성능 평가

Games_20%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.2103 ±.0085		.2462 ±.0088		.2961 ±.0106		.4072 ±.0120	
선행	$n_{aug}=2$	.1394 ±.0020	(−33.7%)	.1781 ±.0013	(−27.7%)	.2171 ±.0033	(−26.7%)	.3373 ±.0009	(−17.2%)
	$n_{aug}=5$	.2212 ±.0032	(+5.2%)	.2577 ±.0030	(+4.7%)	.3082 ±.0028	(+4.1%)	.4215 ±.0017	(+3.5%)
	$n_{aug}=10$	.2617 ±.0046	(+24.4%)	.2965 ±.0046	(+20.5%)	.3519 ±.0064	(+18.8%)	.4599 ±.0062	(+12.9%)
	$n_{aug}=15$	.2822 ±.0010	(+34.2%)	.3168 ±.0008	(+28.7%)	.3781 ±.0015	(+27.7%)	.4851 ±.0009	(+19.1%)
제안	$n_{aug}=2$	.1880 ±.0067	(−10.6%)	.2233 ±.0066	(−9.3%)	.2645 ±.0087	(−10.7%)	.3742 ±.0086	(−8.1%)
	$n_{aug}=5$	.2638 ±.0027	(+25.4%)	.2994 ±.0015	(+21.6%)	.3513 ±.0041	(+18.6%)	.4617 ±.0022	(+13.4%)
	$n_{aug}=10$	.2734 ±.0024	(+30.0%)	.3081 ±.0024	(+25.2%)	.3638 ±.0019	(+22.8%)	.4713 ±.0041	(+15.8%)
	$n_{aug}=15$	.3010 ±.0028	(+43.1%)	.3340 ±.0021	(+35.7%)	.3943 ±.0039	(+33.1%)	.4963 ±.0010	(+21.9%)

표 16-2. 아마존 비디오게임 20% 부분집합 기준 증강 규모 별 성능 평가

Games_30%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3379 ±.0023		.3728 ±.0016		.4419 ±.0026		.5499 ±.0021	
선행	$n_{aug}=2$	.2124 ±.0226	(−37.2%)	.2535 ±.0211	(−32.0%)	.3012 ±.0266	(−31.8%)	.4285 ±.0223	(−22.1%)
	$n_{aug}=5$	.2977 ±.0033	(−11.9%)	.3357 ±.0026	(−9.9%)	.4016 ±.0015	(−9.1%)	.5191 ±.0052	(−5.6%)
	$n_{aug}=10$	.3161 ±.0048	(−10.7%)	.3532 ±.0047	(−9.0%)	.4225 ±.0096	(−8.3%)	.5374 ±.0096	(−3.4%)
	$n_{aug}=15$	.3285 ±.0021	(−2.8%)	.3665 ±.0022	(−1.7%)	.4376 ±.0015	(−1.0%)	.5554 ±.0021	(+1.0%)
제안	$n_{aug}=2$	.2534 ±.0137	(−25.0%)	.2919 ±.00131	(−21.7%)	.3476 ±.0150	(−21.3%)	.4672 ±.0131	(−15.1%)
	$n_{aug}=5$	.3134 ±.0046	(−7.3%)	.3505 ±.0043	(−6.0%)	.4143 ±.0040	(−6.2%)	.5292 ±.0030	(−3.8%)
	$n_{aug}=10$	.3417 ±.0003	(+1.1%)	.3774 ±.0012	(+1.2%)	.4457 ±.0031	(+0.8%)	.5558 ±.0027	(+1.1%)
	$n_{aug}=15$	.3553 ±.0016	(+5.1%)	.3898 ±.0025	(+4.6%)	.4604 ±.0023	(+4.2%)	.5670 ±.0042	(+3.1%)

표 16-3. 아마존 비디오게임 30% 부분집합 기준 증강 규모 별 성능 평가

Games_40%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3638 ±.0023		.3993 ±.0017		.4729 ±.0042		.5834 ±.0023	
선행	$n_{aug}=2$	.2467 ±.0042	(−32.2%)	.2881 ±.0043	(−27.9%)	.3448 ±.0042	(−27.1%)	.4732 ±.0047	(−18.9%)
	$n_{aug}=5$	.3236 ±.0022	(−11.1%)	.3630 ±.0016	(−9.1%)	.4353 ±.0041	(−7.9%)	.5572 ±.0021	(−4.5%)
	$n_{aug}=10$	.3398 ±.0088	(−6.6%)	.3778 ±.0082	(−5.4%)	.4508 ±.0119	(−4.7%)	.5685 ±.0107	(−2.6%)
	$n_{aug}=15$	.3481 ±.0028	(−4.3%)	.3868 ±.0031	(−3.1%)	.4626 ±.0048	(−2.2%)	.5818 ±.0058	(−0.3%)
제안	$n_{aug}=2$	.2835 ±.0038	(−22.1%)	.3247 ±.0052	(−18.7%)	.3918 ±.0053	(−17.1%)	.5192 ±.0097	(−11.0%)
	$n_{aug}=5$	.3420 ±.0020	(−6.0%)	.3792 ±.0011	(−5.0%)	.4514 ±.0026	(−4.5%)	.5663 ±.0009	(−2.9%)
	$n_{aug}=10$	.3665 ±.0028	(+0.7%)	.4026 ±.0019	(+0.8%)	.4777 ±.0030	(+1.0%)	.5895 ±.0040	(+1.0%)
	$n_{aug}=15$	.3810 ±.0018	(+4.7%)	.4161 ±.0013	(+4.2%)	.4919 ±.0034	(+4.0%)	.5964 ±.0043	(+2.2%)

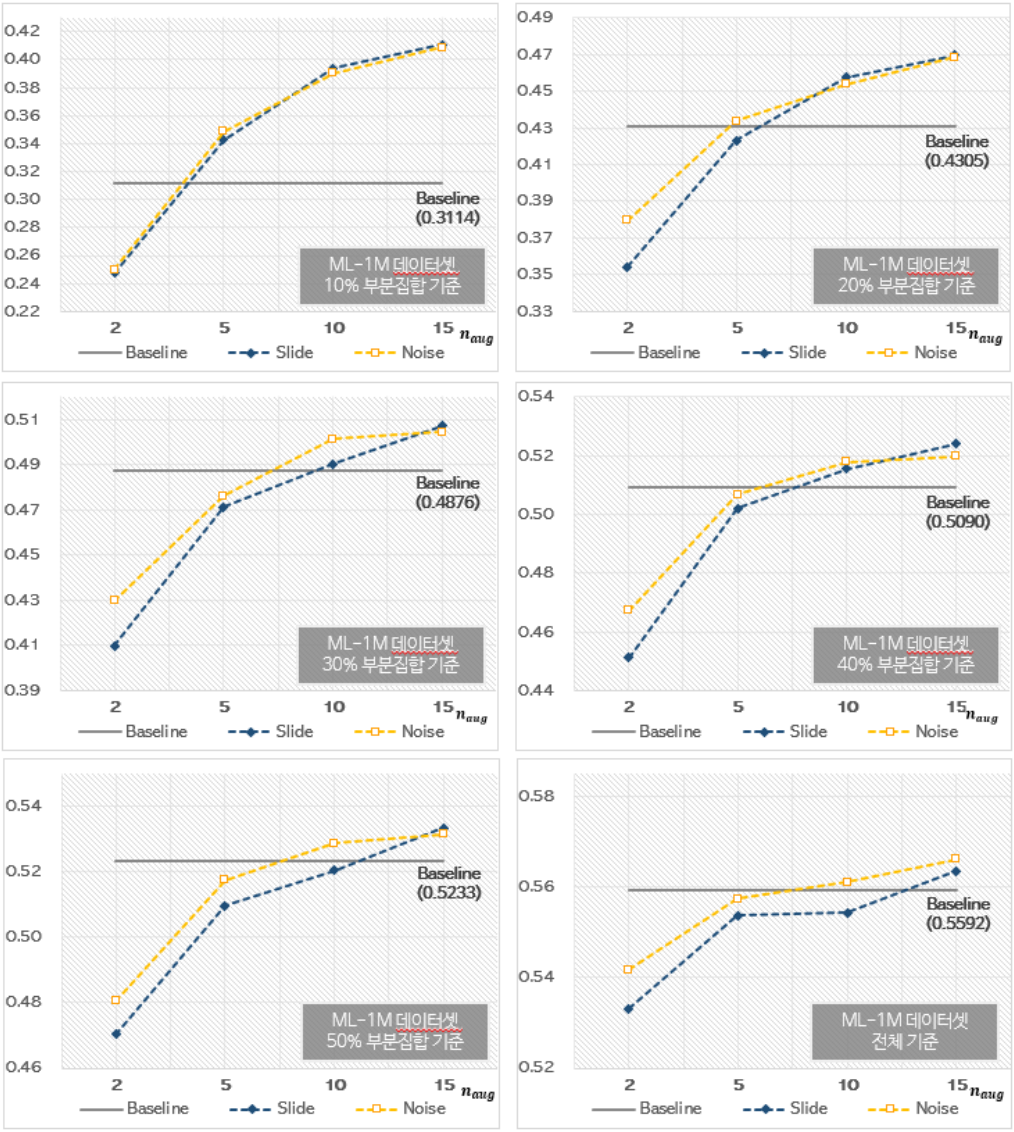
표 16-4. 아마존 비디오게임 40% 부분집합 기준 증강 규모 별 성능 평가

Games_50%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.3922 ±.0044		.4290 ±.0033		.5062 ±.0048		.6197 ±.0034	
선행	$n_{aug}=2$	.2791 ±.0050	(−28.9%)	.3207 ±.0052	(−25.2%)	.3880 ±.0075	(−23.3%)	.5171 ±.0083	(−16.6%)
	$n_{aug}=5$	.3473 ±.0029	(−11.5%)	.3869 ±.0034	(−9.8%)	.4642 ±.0049	(−8.3%)	.5866 ±.0068	(−5.3%)
	$n_{aug}=10$	.3589 ±.0065	(−8.5%)	.3976 ±.0071	(−7.3%)	.4774 ±.0082	(−5.7%)	.5970 ±.0098	(−3.7%)
	$n_{aug}=15$	.3658 ±.0034	(−6.7%)	.4050 ±.0025	(−5.6%)	.4816 ±.0054	(−4.9%)	.6025 ±.0028	(−2.8%)
제안	$n_{aug}=2$	.3115 ±.0030	(−20.6%)	.3522 ±.0023	(−17.9%)	.4211 ±.0031	(−16.8%)	.5470 ±.0018	(−11.7%)
	$n_{aug}=5$	.3653 ±.0032	(−6.9%)	.4032 ±.0028	(−6.0%)	.4818 ±.0024	(−4.8%)	.5990 ±.0014	(−3.3%)
	$n_{aug}=10$	.3931 ±.0016	(+0.2%)	.4293 ±.0018	(+0.1%)	.5095 ±.0022	(+0.7%)	.6212 ±.0030	(+0.2%)
	$n_{aug}=15$	.4031 ±.0028	(+2.8%)	.4392 ±.0022	(+2.4%)	.5211 ±.0038	(+2.9%)	.6328 ±.0019	(+2.1%)

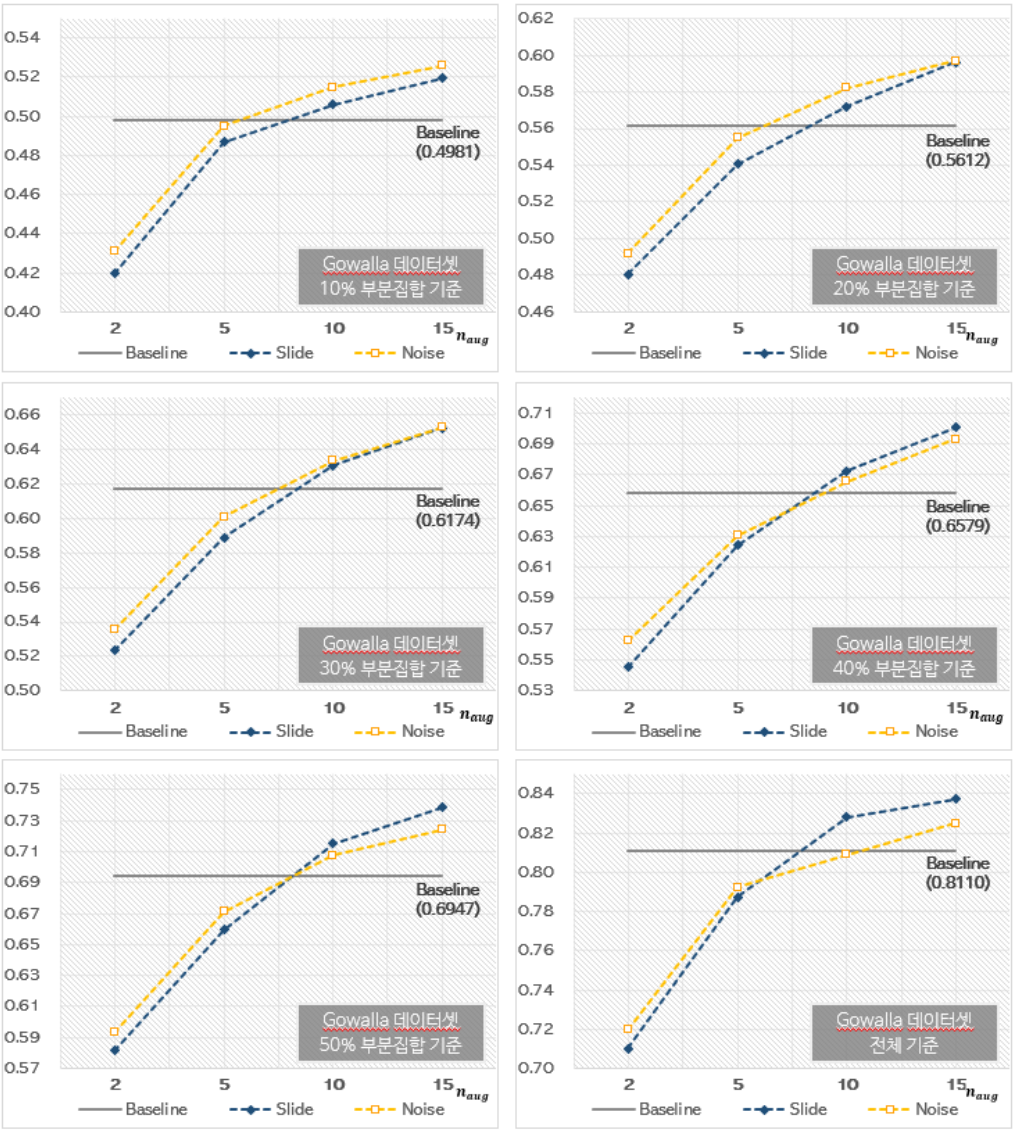
표 16-5. 아마존 비디오게임 50% 부분집합 기준 증강 규모 별 성능 평가

Games_100%		NDCG@5		NDCG@10		HR@5		HR@10	
베이스라인		.4684 ±.0051		.5017 ±.0044		.5941 ±.0034		.6968 ±.0031	
선행	$n_{aug}=2$	.3589 ±.0025	(−23.4%)	.3983 ±.0036	(−20.6%)	.4805 ±.0024	(−19.1%)	.6022 ±.0060	(−13.6%)
	$n_{aug}=5$	.4071 ±.0024	(−13.1%)	.4445 ±.0020	(−11.4%)	.5331 ±.0014	(−10.3%)	.6484 ±.0021	(−6.9%)
	$n_{aug}=10$	.4225 ±.0023	(−9.8%)	.4586 ±.0018	(−8.6%)	.5468 ±.0035	(−8.0%)	.6582 ±.0042	(−5.5%)
	$n_{aug}=15$	.4307 ±.0033	(−8.0%)	.4666 ±.0027	(−7.0%)	.5568 ±.0040	(−6.3%)	.6674 ±.0015	(−4.2%)
제안	$n_{aug}=2$	.3862 ±.0021	(−17.5%)	.4250 ±.0019	(−15.3%)	.5162 ±.0028	(−13.1%)	.6361 ±.0027	(−8.7%)
	$n_{aug}=5$	.4370 ±.0028	(−6.7%)	.4738 ±.0016	(−5.6%)	.5631 ±.0049	(−5.2%)	.6769 ±.0025	(−2.9%)
	$n_{aug}=10$	.4688 ±.0034	(+0.1%)	.5031 ±.0031	(+0.3%)	.5936 ±.0045	(−0.1%)	.6993 ±.0059	(+0.4%)
	$n_{aug}=15$	.4809 ±.0027	(+2.7%)	.5144 ±.0031	(+2.5%)	.6039 ±.0040	(+1.7%)	.7075 ±.0055	(+1.5%)

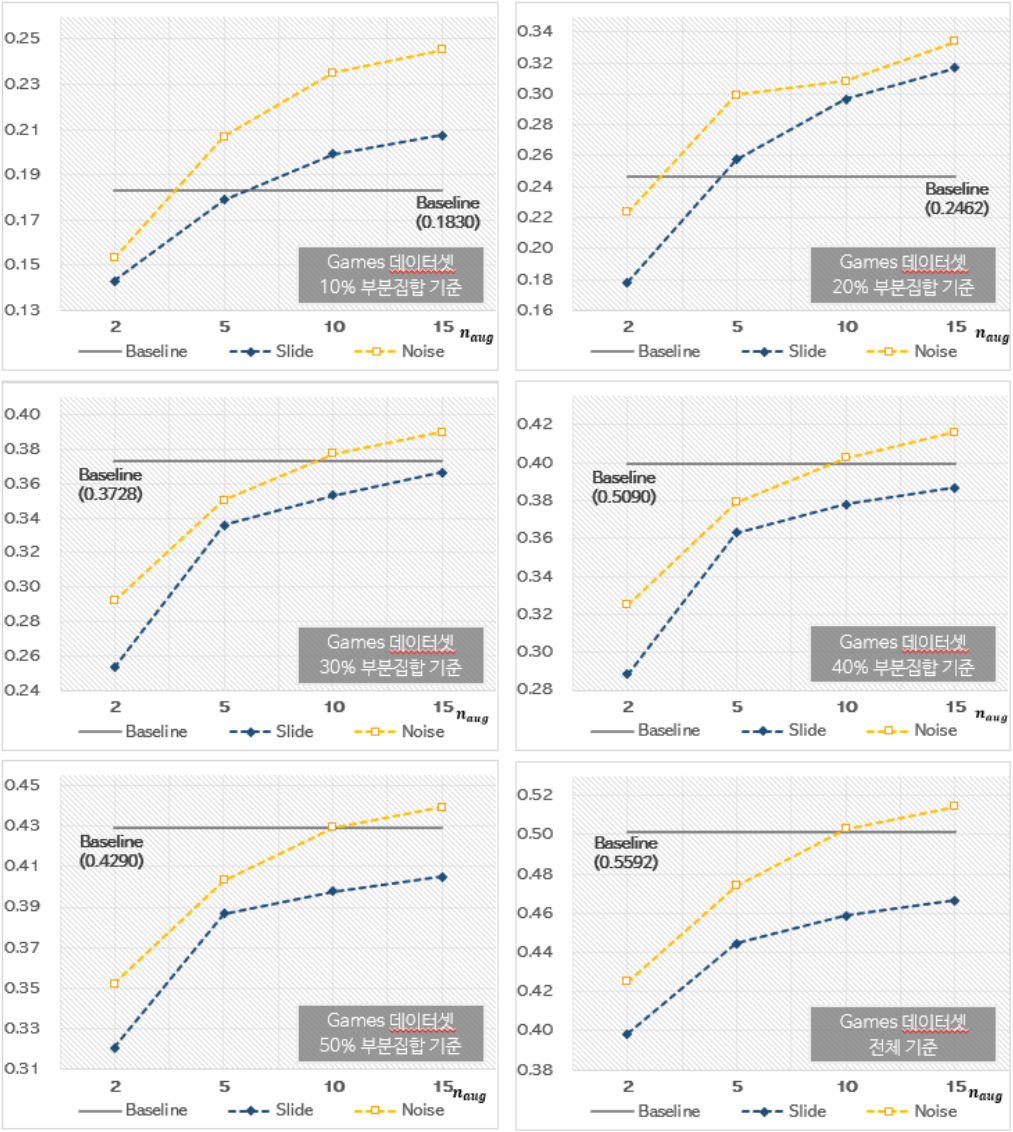
표 16-6. 아마존 비디오게임 전체 데이터셋 기준 증강 규모 별 성능 평가



그래프 2-1. 증강 규모에 따른 무비렌즈 데이터셋 성능 변화 (NDCG@10 기준)



그래프 2-2. 증강 규모에 따른 고왈라 데이터셋 성능 변화 (NDCG@10 기준)



그래프 2-3. 증강 규모에 따른 아마존 비디오게임 데이터셋 성능 변화 (NDCG@10 기준)

제 5 장 논 의

제 1 절 각 데이터 증강 기법의 효과 해석

앞서 상술하였듯 본 연구에서는 기존의 선행 연구를 통해 제안된 2 가지의 데이터 증강 기법과 더불어 본 연구에서 새로이 제안하는 4 가지의 증강 기법을 딥러닝 기반 순차 추천 시스템에 적용해보고 각 기법이 순차 추천 성능에 미치는 영향을 확인하였다. 이어 본 절에서는 각 데이터 증강 기법이 그러한 효과를 가져오게 되는 이유가 무엇인지 추론해보고자 한다.

실험 결과를 통해 본 연구에서 제안한 직접 변형(direct corruption) 방식, 특히 노이즈/중복성 추가 기법이 전반적으로 뛰어난 성능 개선 효과를 나타내는 것을 확인할 수 있었다. 이러한 우수성의 실마리는 아이템 시퀀스 내의 '스킵(skip)' 패턴에서 찾을 수 있을 것이다. 기본적으로 순차 추천 시스템이 아이템 시퀀스에 내재한 시계열 패턴, 곧 순서 정보를 중요하게 고려하는 모델이긴 하나, 모든 직접적인 이웃 관계가 중요한 것은 아닐 것이다. 실제 기존 선행 연구들에 따르면, 마코프체인(MC) 기반 모델에서 바로 직전의 한 아이템만을 고려하는 1 차(1st-order) 마코프체인 모델[10] 보다 이전의 여러 아이템을 함께 고려하는 다차(multi-order) 모델[11] 이 뛰어난 성능을 나타내는 것으로 확인되었으며, 시퀀스 내 개별 아이템 단위(point-wise) 의

변화 패턴 뿐만 아니라 집합 단위(union-level) 변화, 스킵(skip) 패턴 등을 다양하게 고려하는 것이 성능 개선에 기여할 수 있음[35,37] 이 확인된 바 있다. 이처럼 원본 시퀀스가 갖는 아이템 간 순서 관계를 엄격하게 유지하기 보다는 조금 느슨한 관계로서 모델링할 때 더욱 효과적일 수 있다는 기존 연구들의 교훈을 생각한다면, 특히 노이즈/중복성 추가 기법은 추가되는 데이터 샘플을 통해 결과적으로 아이템 시퀀스 내의 스킵 패턴을 학습시키게 하여 성능 개선을 가져오는 것이라 해석할 수 있을 것이다.

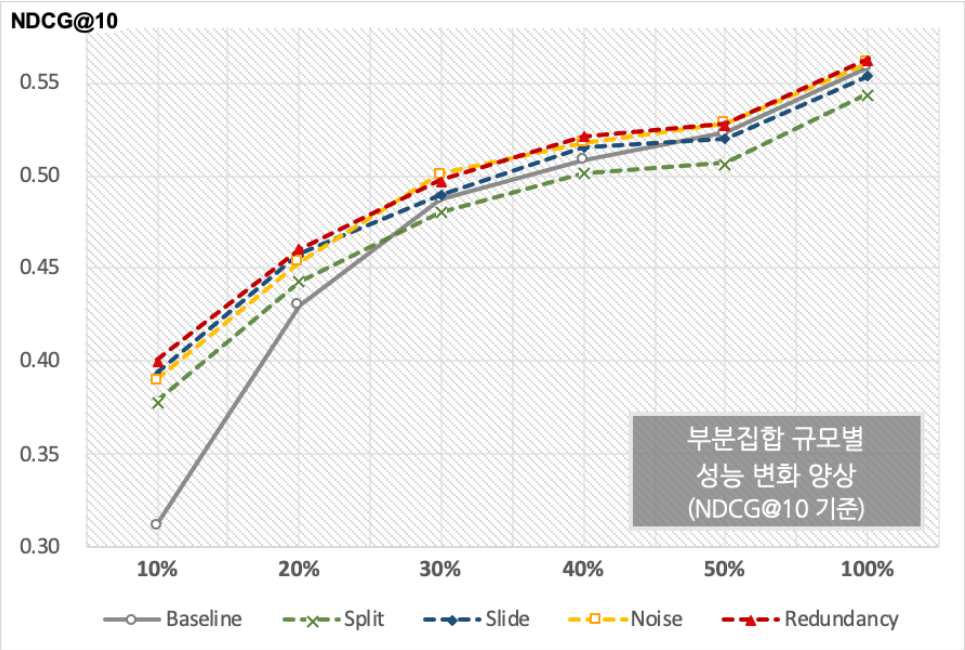
한편 유사 아이템 대체 기법의 경우, '유사한 아이템' 으로 간주되어 원본을 대체한 아이템이 실제로 유사 아이템이라고 볼 수 있을지를 검증하는 과정이 본 실험에서는 포함되어 있지 않다. 또한 워드 임베딩 기법의 하나인 워드투벡터(Word2Vec) 알고리즘을 응용하여 유사 아이템을 계산하여 구하는 과정 역시 정교하게 튜닝된 것이 아니므로, 원본 아이템과 얼마나 유사한지 확인할 수 없다는 점 역시 해당 기법이 상대적으로 낮은 성능을 가져오는 결과를 설명할 수 있는 하나의 요인이 될 수 있을 것이다. 뿐만 아니라, 유사품은 결국 원본과는 다른 아이템이기에 근본적으로 유사품이 원본을 대체할 수 없는 경우도 존재할 것이다. 구체적으로, 소비자에 따라 특정 제품에 대해 높은 충성도(loyalty) 를 가질 수 있으며, 이러한 경우 해당 제품을 유사한 제품으로 대체하는 것은 명확한 노이즈로서 해당 소비자가 갖는 기존의 선호 특성을 해치는 결과를 가져올 것이다.

선행 연구에서 제안되었던 2 가지 기법들 역시 적용되는 데이터셋에 따라 추천 성능에 미치는 효과가 다르게 나타나나,

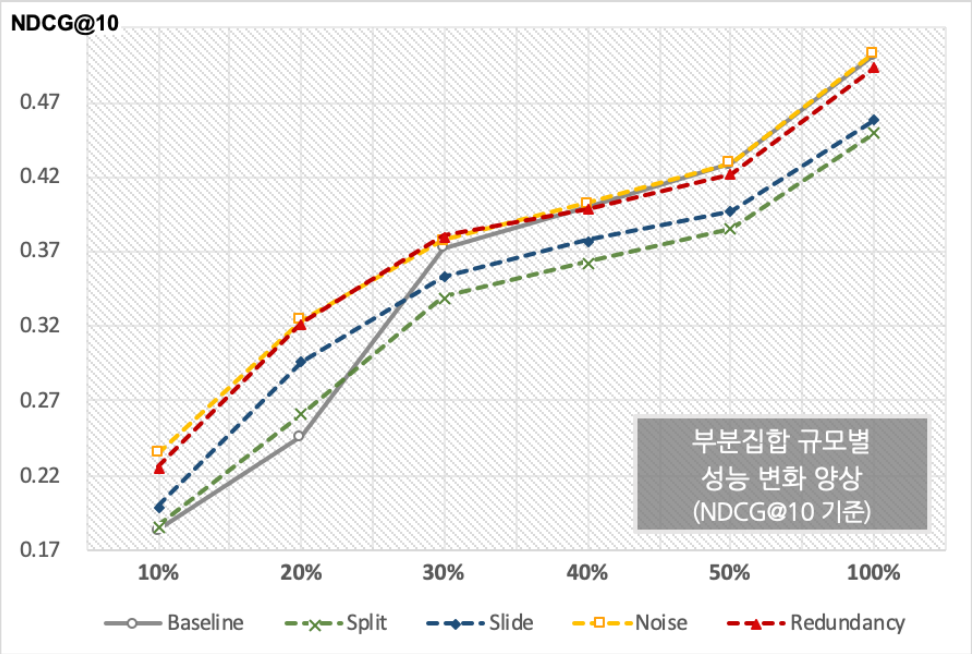
전반적으로 부분집합 분할 기법 대비 슬라이딩 윈도우 기법이 안정적으로 성능 향상에 기여하는 것으로 확인되었다. 특히 고왈라 데이터셋의 경우 전체의 40% 이상을 사용한 부분집합에 대해서는 슬라이딩 윈도우 기법을 적용하였을 때 가장 성능 향상 효과가 큰 것으로 나타났다. 부분집합 분할 기법의 경우 다르게 생각하면 원본 시퀀스에 대해 랜덤하게 다른 비율로 아이템 마스킹을 적용한 것이라고도 볼 수 있다. 아이템 마스킹은 기존의 데이터를 충분히 활용할 수 없게 만들어 결과적으로 원본 대비 활용 가능한 정보가 줄어든다는 점에서, 공통적으로 원본 시퀀스의 부분집합을 활용하는 방식임에도 슬라이딩 윈도우 기법 대비 낮은 개선 효과를 가져오는 결과를 설명할 수 있을 것이다. 나아가, 본 연구에서 제안하는 기법들 중 노이즈/중복성 추가 계열에 비해 상대적으로 아이템 마스킹 기법의 성능 개선 효과가 낮은 것도 이와 같은 맥락에서 이해할 수 있다.

제 2 절 데이터 증강 효과의 포화(saturation)

앞서 데이터 증강 규모(n_{aug})와 데이터 증강 효과의 상관 관계를 살펴본 결과, 데이터 증강을 적용하는 데이터셋 규모가 작을 때에는 적은 규모의 데이터 증강으로도 큰 폭의 성능 개선을 얻을 수 있으나, 데이터셋의 규모가 커질수록 증강을 통한 성능 개선 폭이 작아지는 한편 유의미한 성능 개선 효과를 기대하기 위해서는 보다 큰 규모의 데이터 증강을 필요로 한다는 것을 알 수 있었다. 다시 말해, 데이터가 충분한 규모로 확보됨에 따라 데이터 증강을 적용해도 성능 개선 효과가 점차 희미해지는, 이른바 포화 상태 (saturation) 에 이르게 되는 것이다. 결과적으로 데이터가 충분히 많은 상황이라면 데이터 증강으로 인한 성능 개선의 효율이 높지 않다고 할 수 있다.



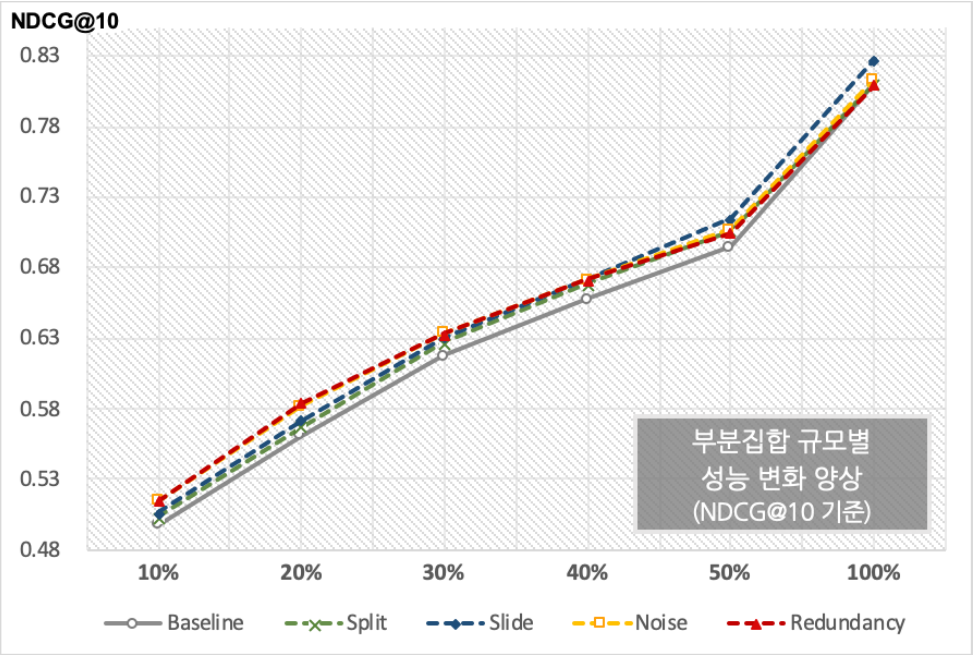
그래프 3-1. 무비렌즈 데이터셋 부분집합 규모별 성능 변화 양상



그래프 3-2. 아마존 비디오게임 데이터셋 부분집합 규모별 성능 변화 양상

[그래프 3-1~3-3] 은 데이터 증강 규모를 일정하게 (n_{aug} = 10) 유지한 채 각 데이터셋의 부분집합 규모를 점차 증가시킴에 따라 베이스라인 모델 및 각 데이터 증강 기법 적용 시의 성능이 어떻게 변화해가는지 그 양상을 나타낸 것이다.

무비렌즈(ML-1M) 와 아마존 비디오게임(Games) 데이터셋의 경우 부분집합의 규모가 전체 데이터셋에 해당하는 100% 를 향해감에 따라 최적 기법조차 베이스라인 성능과 크게 다르지 않은 범위로 수렴해가지만, 전체의 10~20% 에 수준에서는 대부분의 기법이 뚜렷한 성능 개선 효과를 가져오는 것을 확인할 수 있다. 고왈라(Gowalla) 데이터셋의 경우 부분집합 규모에 따른 차이가 크지 않으나, 전체 데이터셋을 모두 사용한 경우는 확실하게 성능 포화 상태에 이르는 것을 알 수 있다.



그래프 3-3. 고왈라 데이터셋 부분집합 규모별 성능 변화 양상

이러한 결과를 바탕으로 실제 순차 추천 시스템에 데이터 증강을 적용하고자 할 때의 시사점을 이끌어 낼 수 있다. 추천 시스템의 경우 일반적으로 다루어지는 데이터가 실제 사용자의 행동 로그(log) 라는 특성상 타 분야에 비해 추가적인 데이터 확보가 더욱 어려운 편이다. 특히 시스템 초창기에 작은 규모의 데이터셋으로는 높은 성능의 추천 시스템을 구현하는 데에 한계가 존재하며, 추가적인 데이터를 충분히 확보할 때까지 상대적으로 낮은 성능으로 인해 사용자 만족도 저하를 감수할 수밖에 없다. 하지만 순차 추천 시스템이 적용되는 서비스 초기, 데이터를 충분히 확보하지 못한 상황에서 데이터 증강을 적용한다면 단순히 사용자의 추가적인 활동 로그를 확보하기 위해 비용을 들이는 것에 비해 훨씬 효율적으로 성능 향상을 도모할 수 있을 것이다.

제 3 절 SOTA 모델의 추가 개선 가능성

앞서 3 장의 3 절에서 상술하였듯 본 연구에서 베이스라인으로 삼은 모델 *SASRec*[32] 은 2018 년에 제안된 것으로, 현재 기준의 SOTA(State-Of-The-Art) 성능의 모델은 아니다. 이에 본 연구의 실험에서 사용된 각 데이터셋[6]에 대해 현재 SOTA 모델의 성능과 데이터 증강을 통해 향상된 성능을 비교해보았으며, 해당 결과를 아래의 [표 17] 와 같이 정리하였다.

데이터셋	무비렌즈		아마존 비디오게임	
	NDCG@10	HR@10	NDCG@10	HR@10
SASRec[7]	0.5592	0.7929	0.5017	0.6968
SASRec + DataAug[8]	0.5662	0.7961	0.5144	0.7075
TiSASRec	0.5706	0.8038	0.4797	0.7087
SOTA 대비 성능	99.23%	99.04%	107.23%	99.83%

표 17. 데이터셋 별 데이터 증강 결과와 SOTA 성능의 비교

[6] 고왈라 데이터셋의 경우 TiSASRec[34] 연구에서는 활용되지 않았으며, 본 연구와 동일한 지표(metric) 를 사용하여 성능을 평가한 선행 연구를 발견하지 못하여 우선 비교 대상에서 제외하였음

[7] 본 연구에서 실험을 통해 재구현(reproduce) 한 성능 기준. 참고로 TiSASRec 논문을 통해 보고된 성능은 다음과 같음. ML-1M: 0.5024 / 0.7929 , Games: 0.4714 / 0.6952

[8] 해당 데이터셋에 대해 최고 성능을 나타낸 '노이즈 추가' 기법을 n_{aug} =15 규모로 적용한 결과

표를 통해 알 수 있듯 무비렌즈, 아마존 비디오게임 데이터셋에 대해 데이터 증강 적용 시의 성능은 현재 SOTA 모델 중 하나인 *TiSASRec*[34] 의 성능과 유사한(competitive) 수준인 것으로 나타났다.

이처럼 데이터 증강이 모델 아키텍처를 변형하는 것과 유사한 수준으로 성능 향상을 가져올 수 있다고 해석하더라도, 대규모 데이터셋에 대해 대규모의 데이터 증강을 적용할 경우 학습 데이터셋이 늘어나는 만큼 모델 학습에 추가적인 시간이 소요될 것이다. 따라서 본 연구에서는 궁극적으로 딥러닝 기반 순차 추천 모델의 성능 향상이라는 목표를 이루기 위해 '데이터 증강'을 '모델 아키텍처 변형'을 대체(replace) 하는 접근법으로 보는 것이 아니라, 두 접근법이 상호보완적으로 활용될 수 있음을 강조하고자 한다. 데이터 증강은 기본적으로 모델의 아키텍처를 변화시키지 않는다는 점에서, 현재 제시되어 있는 다양한 SOTA 모델에 대해서도 본 연구에서 제안하는 데이터 증강 기법을 적용한다면 추가적인 성능 개선이 나타날 가능성이 존재하는 것이다. 실제 *TiSASRec* 을 포함한 여러 SOTA 모델에 대해 실험을 진행하여 이를 실증적으로 뒷받침하는 연구는 본 연구의 한계를 보완하기 위한 향후 과제라고 할 수 있을 것이다.

제 4 절 데이터셋 특성에 따른 적용 가이드라인

마지막으로 본 절에서는 데이터셋의 특성에 따라 데이터 증강의 효과가 어떻게 다르게 나타나는지에 대해 서술하고자 한다. 본 연구에서는 추천에 사용되는 데이터셋이 지니는 핵심적인 특성을 1) 데이터셋의 규모, 2) 희소 정도 (sparsity), 3) 평균 시퀀스 길이라 보고, 이들 각 특성에 따라 데이터 증강 효과에 있어 어떠한 차이가 존재하는지 그 상관 관계를 확인하고자 하였으며, 이를 통해 최종적으로 데이터셋의 특성에 따라 가장 적합한 데이터 기법을 제안하는 것을 목표로 하였다. 이를 위해 각 특성의 측면에서 차별성을 지니는 세 개의 데이터셋에 대해 실험을 진행하였으며, 이들 데이터셋에 대한 특성 정보는 아래 [표 18] 를 통해 확인할 수 있다.

데이터셋	규모(size)[9]	희소 정도(sparsity)	평균 시퀀스 길이[10] (avg. sequence length)
무비렌즈(ML-1M)	Small (6,040)	Low (95.16)	Long (163.50)
고왈라(Gowalla)	Large (85,034)	High (99.98)	Long (52.83)
아마존 비디오게임 (Games)	Large (64,073)	High (99.97)	Short (6.88)

표 18. 사용된 데이터셋에 대한 특성 정보

[9] 데이터 수, 곧 세션(session) 수를 의미하는 것으로 간주함
[10] 세션 당 평균 아이템 수

하지만 앞서 확인한 실험 결과를 통해 알 수 있듯이, 데이터셋에 따라 증강 효과의 양상은 각기 다르게 나타났으며 데이터셋의 규모/ 희소 정도/ 평균 시퀀스 길이 등의 특성에 따라 데이터 증강 효과의 측면에서 공유되는 경향성이 뚜렷하게 나타나지는 않는 것으로 확인되었다. 또한 공통적으로 데이터 증강 효과가 크게 발휘되는 임계점이 되는 규모는 존재하지 않는 것으로 보이며, 데이터셋에 따라 성능이 포화 상태에 이르는 양상 및 타이밍이 각기 다르게 나타났다. 다만 본 연구에서는 3 개의 제한적인 경우에 대해서만 실험을 진행하였기 때문에, 더 많은 데이터셋에 대해 포괄적인 실험을 수행해본다면 본 연구에서는 미처 드러나지 않았던 어떠한 경향성을 발견하게 될 수도 있겠으나, 현 시점의 결과를 통해서는 확언할 수 없는 것으로 보인다. 이는 결국, 추천 시스템에서 사용되는 데이터셋은 단순히 규모 / 희소 정도 / 평균 시퀀스 길이 등의 요소로 환원되지 않는 복잡한 패턴(characteristics) 를 지니고 있다고 이해할 수도 있을 것이다.

결국 본 연구에서 직접 실험을 적용해 본 벤치마크 데이터셋 외에 완전히 새로운 데이터셋을 대상으로 추천 시스템을 운영해야 하는 실제 현실의 상황에서는 어느 정도 규모 (n_{aug}) 로 데이터 증강을 적용해야 성능 개선 효과가 있을지 명확하지 않으며, 이는 본 연구가 지니는 한계점이라 할 수 있다. 단순한 방법이지만, 임의의 데이터셋이 데이터 증강을 적용하였을 때 성능 개선 효과가 있을지 알아보기 위해서는 일단 적용해보고 그 결과를 확인해보는 수밖에 없는 것으로 보인다. 다만, 본 연구에서 제안한 노이즈/중복성 추가 기법은 대부분의

데이터셋에서 유의한 성능 개선 효과를 나타냈으므로, 해당 기법을 우선적으로 적용하는 것을 고려해볼 수 있을 것이다.

제 6 장 결 론

본 연구는 기존에 순차 추천 분야에서 충분히 연구되지 않았던 데이터 증강에 초점을 맞추어, 이를 통해 학습 데이터가 충분하지 않은 저자원(low-resource) 상황에서 딥러닝 기반 모델의 추천 성능을 개선할 수 있음을 확인하였다. 선행 연구에서 제안되었던 데이터 증강 기법들이 공통적으로 원본 데이터의 부분집합 (subsequence) 를 활용하는 접근법을 취한 것과 달리, 본 연구에서 제안한 방식은 원본 시퀀스에 직접적인 변형(direct corruption) 을 가하는 것으로서, 복수의 데이터셋에 대한 정량 실험을 통해 이들 기법이 기존 접근법 대비 더 높거나 유사한(competitive) 수준의 성능을 나타내는 것을 확인할 수 있었다.

또한, 데이터 증강은 기본적으로 데이터 전처리 과정에서 이루어지며 모델의 아키텍처를 변화시키지 않는다는 점에서, 본 연구에서 제안하는 데이터 증강 기법을 통해 현재 제시되어 있는 다양한 SOTA(State-Of-The-Art) 모델의 성능을 추가적으로 개선시킬 수 있을 것이는 가능성을 제시하였다. 이전에 순차 추천에서 데이터 증강을 다룬 선행 연구의 경우 특정 데이터셋에 대해서만 제한적으로 적용해보고 효과를 확인하였으나, 본 연구에서는 선행 연구에서 제안한 접근법을 포함하여 다양한 데이터 증강 방식을 여러 데이터셋에 대해 포괄적으로 적용해 보았다. 이를 통해 각 증강 방식에 따른 성능

변화의 차별적 양상을 확인할 수 있었으며, 나아가 데이터 증강이 추천 시스템 디자인에 있어 보편적인 전처리 기법의 하나로 기능할 수 있을 가능성을 검증하였다.

한편, 기존의 순차 추천 분야에서 아직 충분히 연구되지 않은 데이터 전처리 측면에서의 개선, 구체적으로 데이터 증강에 주목하여 이에 대해 깊이 있게 다룬 하나의 연구로서 향후 관련 연구들을 발전시키는 과정에서 기초 자료로 활용 가능할 것이다.

하지만 현재까지 제안되어 온 다양한 딥러닝 기반 순차 추천 모델 가운데 제한된 하나의 모델에 대해서만 적용해 보았다는 점은 본 연구의 한계로서 지적할 수 있다. 이에 추후 다른 네트워크 아키텍처를 기반으로 하는 모델들에 대해서도 확장이 가능할지 추가적으로 확인하는 연구를 진행할 수 있을 것이다.

참고 문헌

[1] H. Fang, D. Zhang, Y. Shu, and G. Guo, 2020, “Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations”, *ACM Trans. Inf. Syst.* 39, 1, Article 10 (Nov 2020). 42 pages.

[2] C. Shorten, T. M. Khoshgoftaar, 2019, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*. 6, 60 (2019).

[3] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q. V. Le, 2019, “AutoAugment: Learning Augmentation Strategies From Data,” In *CVPR*. IEEE, 113-123.

[4] D.S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E.D. Cubuk, Q. V. Le, 2019, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” In *Interspeech*, 2613-2617.

[5] J. Wei, K. Zou, 2019, “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks,” In *EMNLP-IJCNLP*. ACL.

[6] J. Leskovec, A. Rajaraman, and J. Ullman, 2020, “Mining of Massive Datasets (3rd Edition)”, *Cambridge University Press*.

[7] Y. Hu, Y. Koren and C. Volinsky, 2008, “Collaborative Filtering for Implicit Feedback Datasets”, In *ICDM*. IEEE, 263-272.

[8] Y. Koren, R. Bell and C. Volinsky, 2009, "Matrix Factorization Techniques for Recommender Systems," In *Computer,* vol. 42(8), 30-37.

[9] S. Rendle, 2010, "Factorization Machines," In *ICDM*. IEEE, 995-1000.

[10] S. Kabbur, X. Ning, and G. Karypis, 2013, “FISM: factored item similarity models for top-N recommender systems,” In *KDD*. ACM, 659-667.

[11] O. Barkan, N. Koenigstein, E. Yogev, and O. Katz, 2019, "CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations," In *RecSys*. ACM, 228-236.

[12] R. Salakhutdinov, A. Mnih, and G. Hinton, 2007, "Restricted Boltzmann machines for collaborative filtering," In *ICML*. ACM, 791-798.

[13] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T-S. Chua, 2017. "Neural Collaborative Filtering," In *WWW*, 173-182.

[14] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, 2017, "Deepfm: a factorization-machine based neural network for ctr prediction," In *IJCAI*, 1725-1731.

[15] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, 2017, "What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation," In *WWW*, 391-400.

[16] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, 2016, "Convolutional Matrix Factorization for Document Context-Aware Recommendation," In *RecSys*. ACM, 233-240.

[17] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, 2015, "AutoRec: Autoencoders Meet Collaborative Filtering," In *WWW*, 111-112.

[18] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, 2016, "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems," In *WSDM*. ACM, 153-162.

[19] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, 2018, "Variational Autoencoders for Collaborative Filtering," In *WWW*. 689-698.

[20] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, 2019, "Sequential Variational Autoencoders for Collaborative Filtering," In *WSDM*, ACM. 600-608.

[21] S. Wang, L. Cao, and Y. Wang, 2019, "A Survey on Session-based Recommender Systems," *arXiv* preprint arXiv:1902.04864.

[22] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. 2010, "Factorizing personalized markov chains for next-basket recommendation," In *WWW*, 811-820.

[23] R. He and J. McAuley, 2016, "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation," In *ICDM*, IEEE.

[24] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, 2015, "Session-based recommendations with recurrent neural networks," *arXiv* preprint arXiv:1511.06939.

[25] B. Hidasi and A. Karatzoglou, 2018, "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations," In *CIKM*. ACM, 843-852.

[26] C. Xu, P. Zhao, Y. Liu, J. Xu, V. Sheng, Z. Cui, X. Zhou, and H. Xiong, 2019, "Recurrent Convolutional Neural Network for Sequential Recommendation," In *WWW*, 3398-3404.

[27] F. Zhou, Z. Wen, K. Zhang, G. Trajcevski, and T. Zhong, 2019, "Variational Session-based Recommendation Using Normalizing Flows," In *WWW*, 3476-3475.

[27] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, 2017, "Neural Attentive Session-based Recommendation," In *CIKM*. ACM, 1419-1428.

[28] L. Yu, C. Zhang, S. Liang, and X. Zhang, 2019, "Multi-Order Attentive Ranking Model for Sequential Recommendation," In *AAAI*. 5709-5716.

[29] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, 2018, "STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation," In *KDD*. ACM, 1831-1839.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, 2017, "Attention is all you need," In *NIPS*. 6000-6010.

[31] J. Devlin, M-W. Chang, K. Lee, K. Toutanova, 2018, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv* preprint arXiv:1810.04805.

[32] W-C. Kang, and J. McAuley, 2018, "Self-Attentive Sequential Recommendation." In *ICDM*. IEEE, 197-206.

[33] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, 2019, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer," In *CIKM*. ACM, 1441-1450.

[34] J. Li, Y. Wang, and J. McAuley, 2020, "Time Interval Aware Self-Attention for Sequential Recommendation," In *WSDM*. ACM, 322-330.

[35] J. Tang, and K. Wang, 2018, "Personalized top-n sequential recommendation via convolutional sequence embedding," In *WSDM*. ACM, 565-573.

[36] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, X. He, 2019, "A Simple Convolutional Generative Network for Next Item Recommendation," In *WSDM*. ACM, 582-590.

[37] A. Yan, S. Cheng, W-C. Kang, M. Wan, J. McAuley, 2019, "CosRec: 2D Convolutional Neural Networks for Sequential Recommendation," In *CIKM*. ACM, 2173-2176.

[38] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, 2019, "Session-Based Recommendation with Graph Neural Networks," In *AAAI*. 346-353.

[39] R. Qiu, J. Li, Z. Huang, and H. Yin, 2019, "Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks," In *CIKM*. ACM, 579-588.

[40] O. Vinyals, S. Bengio, and M. Kudlur, 2016, "Order Matters: Sequence to sequence for sets," In *ICLR*.

[41] I. Goodfellow, Y. Bengio and A. Courville, 2016, "Deep Learning," *MIT Press.*

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 2012, “ImageNet classification with deep convolutional neural networks,” In *NIPS*, 1097–1105.

[43] P. Y. Simard, D. Steinkraus and J. C. Platt, 2003, "Best practices for convolutional neural networks applied to visual document analysis," In *ICDAR*. IEEE, 958-963.

[44] J. Lemley, S. Bazrafkan and P. Corcoran, 2017, "Smart Augmentation Learning an Optimal Data Augmentation Strategy," In *IEEE Access*, 5858-5869.

[45] T. Tran, T. Pham, G. Carneiro, L. Palmer, I. Reid, 2017, “A Bayesian Data Augmentation Approach for Learning Deep Models,” In *NIPS*.

[46] T. DeVries, G. W. Taylor, 2017, “Dataset Augmentation in Feature Space,” *arXiv* preprint arXiv:1702.05538.

[47] X. Zhu, Y. Liu, Z. Qin, J. Li, 2017, “Data Augmentation in Emotion Classification Using Generative Adversarial Networks,” *arXiv* preprint arXiv:1711.00648.

[48] N. Kanda, R. Takeda and Y. Obuchi, 2013, "Elastic spectral distortion for low resource speech recognition with deep neural networks," In *ASRU*. IEEE, 309-314.

[49] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng, 2014, “Deep Speech: Scaling up end-to-end speech recognition,” *arXiv* preprint arXiv:1412.5567.

[50] T. Ko, V. Peddinti, D. Povey, S. Khudanpur, 2015, “Audio augmentation for speech recognition,” In *Interspeech*, 3586-3589.

[51] S. H. Mallidi and H. Hermansky, 2016, "Novel neural network based fusion for multistream ASR," In *ICASSP*. IEEE, 5680-5684.

[52] R. Sennrich, B. Haddow, A. Birch, 2016, “Improving Neural Machine Translation Models with Monolingual Data,” In *ACL*. 86-96.

[53] Z. Xie, S. I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, A. Y. Ng, 2017, "Data Noising as Smoothing in Neural Network Language Models," In *ICLR*.

[54] W. Y. Wang, D. Yang, 2015, "That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets," In *EMNLP*. ACL.

[55] Q. Wang, H. Yin, H. Wang, Q. Nguyen, Z. Huang, and L. Cui, 2019, "Enhancing Collaborative Filtering with Generative Aug-mentation," In *KDD*. ACM.

[56] M. Wölbitsch, S. Walk, M. Goller, and D. Helic, 2019, "Beggars Can't Be Choosers: Augmenting Sparse Data for Embedding-Based Product Recommendations in Retail Stores," In *UMAP*. ACM.

[57] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, 2015, "E-commerce in Your Inbox: Product Recommendations at Scale," In *KDD*. ACM.

[58] C. Yan, Y. Chen and L. Zhou, 2019, "Differentiated Fashion Recommendation Using Knowledge Graph and Data Aug-mentation," In *IEEE Access,* vol. 7.

[59] Y. K. Tan, X. Xu and Y. Liu, 2016, "Improved Recurrent Neural Networks for Session-based Recommendations," In *DLRS*. ACM, 17-22.

Abstract

A Study on the Improvement of Sequential Recommender System through Data Augmentation

Joo-yeong Song
Department of Intelligence and Information
Graduate School of Convergence Science and Technology
Seoul National University

Recommender systems, which have been developed based on the maturity of the information society, have recently achieved high performance improvement with the development of 'Deep Learning' technology that makes use of artificial deep neural network. Owing to the improved performance, its influence as well as its importance have been increasing today. Generally, it is known that the recommender models based on neural network structure can achieve higher performance than those with traditional approaches, but it premises a large volume of data in order to estimate numerous parameters of the model. However, it is quite expensive to obtain additional data when the size of dataset being in hand is small. Especially, recommender systems

have more difficulty in obtaining additional data compared to other areas in that they deal with actual user behavior logs.

In the field of Computer Vision, where Deep Learning technology is being actively applied, it has been seeking to improve the performance of the model by greatly increasing the volume of training data using various data augmentation technologies. It is also expected that recommender systems can have significant benefits using these data augmentation technologies, but the research on data augmentation has not been sufficiently progressed yet in the recommender systems, especially in the field of sequential recommendation.

Motivated by this situation, we aim to show that various data augmentation techniques can improve the performance of sequential recommender system, especially when the training dataset is small. To this end, we describe how data augmentation changes the performance with the extensive experiment based on the latest sequential recommender model of neural network architecture. Our suggested data augmentation techniques are 1) Noise Injection, 2) Redundancy Injection, 3) Masking, 4) Synonym Substitution, all of which transform original item sequences in the way of direct corruption. Experiments performed with three benchmark datasets – 'Movielens-1M', 'Gowalla' and 'Amazon Video Games' – demonstrate that overall performance improvement can be found when our suggested data augmentation techniques applied. It's notable that the performance improvement can

be large if the size of dataset is relatively small. It can be said that applying data augmentation techniques can be effective to boost the performance, especially when the sequential recommender system doesn't have enough dataset on the early stage of a service.

The contributions of this study can be summarized as follows: 1) it has confirmed with quantitative experiments that data augmentation technology can improve sequential recommendation performance when training dataset is small, 2) it suggests the possibility of further performance improvement of other current SOTA(State-Of-The-Art) models, in that data augmentation is applied in the pre-processing step and does not change the overall model architecture, 3) it has described how the performance changes in different manner according to the extensive application of various data augmentation techniques, and it furthermore verified that data augmentation can work as an universal pre-processing technique in the design of recommender system, 4) it can be referred as a basic research result in the process of developing the research on data augmentation in recommender systems, which has not yet been fully investigated.

Keywords : Sequential Recommendation, Recommender System, Data Augmentation, Data pre-processing, Deep Learning

Student Number : 2019-21923