과제명: NCF with Side Information 확장 실험

학번: U2024041

이름: 최우연

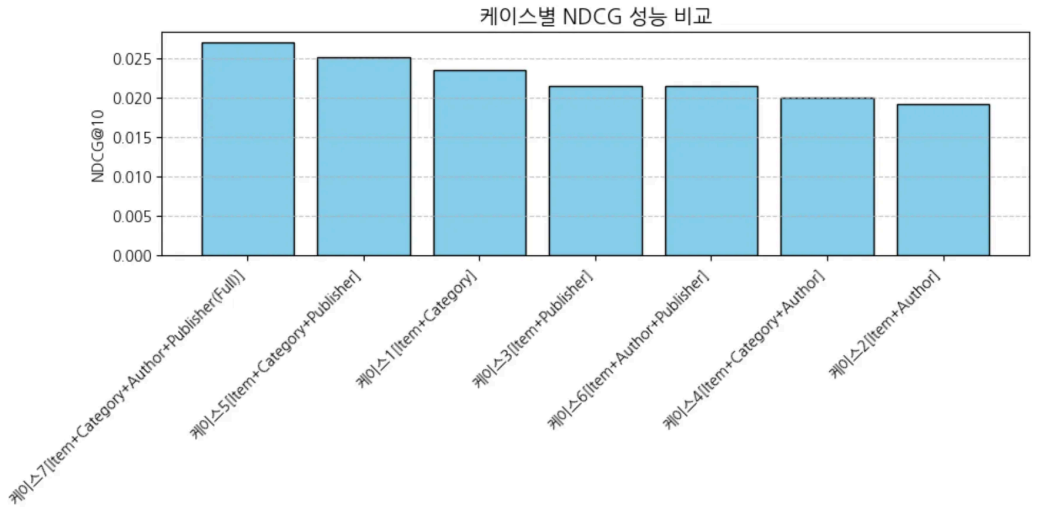
01. 실험 1 - Addition 방식 조합 분석 결과

[실험 결과]

- 7가지 조합의 NDCG@10 비교 표

```
============================================================
📊 케이스별 성능 비교표 (내림차순 정렬)
============================================================
                                          케이스  NDCG@10
케이스7[Item+Category+Author+Publisher(Full)] 0.027004
              케이스5[Item+Category+Publisher] 0.025196
                        케이스1[Item+Category] 0.023576
                       케이스3[Item+Publisher] 0.021515
                 케이스6[Item+Author+Publisher] 0.021466
                  케이스4[Item+Category+Author] 0.020070
                          케이스2[Item+Author] 0.019194
```

- 7가지 조합의 NDCG@10 비교 챠트



[분석]

- 최적의 Side Info 결과 : 케이스7[Item+Category+Author+Publisher(Full)]
- 단일 vs 다중 모델 조합 효과 비교 분석
  - 단일 모델 NDGC@10 은 0.0238
  - 케이스7의 다중모델 조합(Item+Category+Author+Publisher(Full))은 NDCG@10은 단일 모델 대비 +13.5% 성능향상 - 단순 노이즈(무작위 성능 변동)로 보기 어려운 수준의 성능 향상
- 최적 조합 선정 근거
  - 단순히 피처 수가 많다고 항상 성능이 오르진 않지만, 단일모델 대비 케이스7은 “Author + Publisher + Category”를 동시에 활용하면서 가장 안정적이고 높은 성능(+13.5%)을 달성.
  - 이는 피처 간 상호보완적 관계가 존재함을 시사하며, 조합효과(combination effect)가 명확히 관찰됨

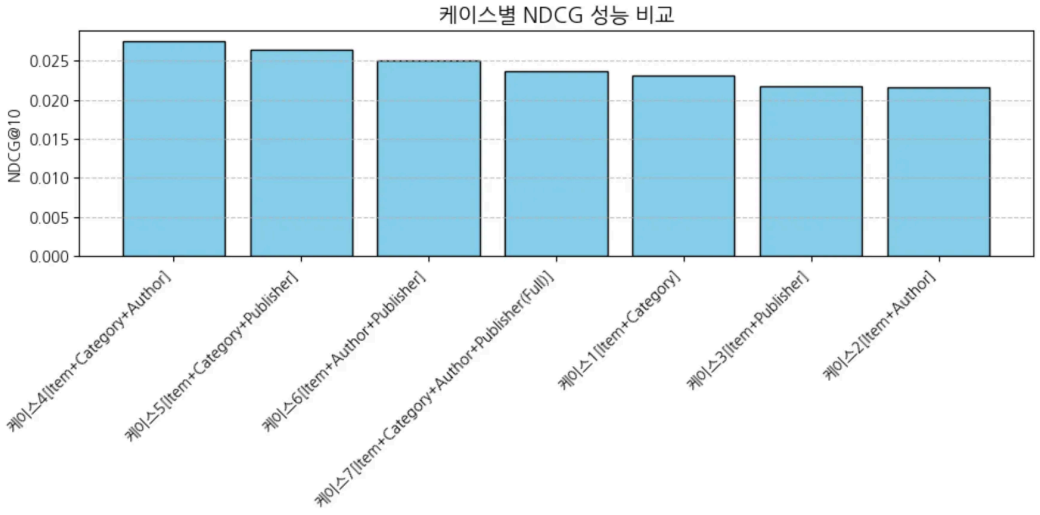
02. 실험 2 - Concatenation 방식 조합 분석

[실험 결과]

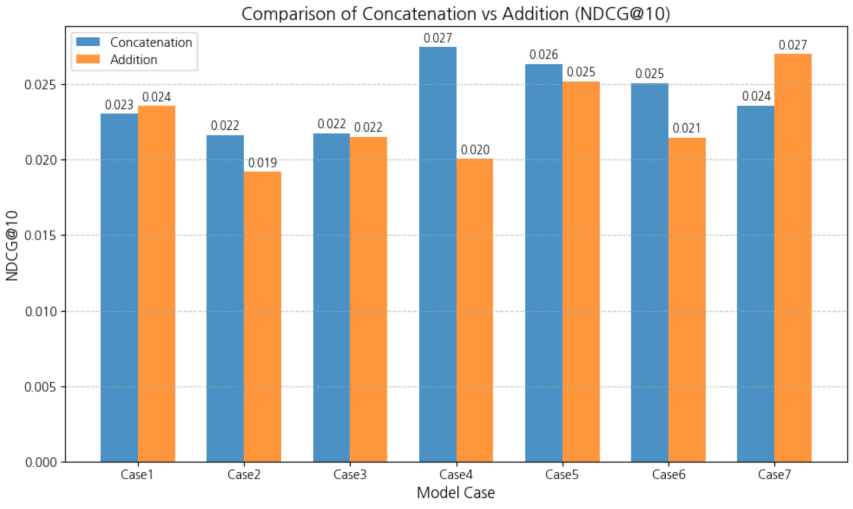
- 7가지 조합의 NDCG@10 비교 표

```
============================================================
📊 케이스별 성능 비교표 (내림차순 정렬)
============================================================
                                          케이스  NDCG@10
                  케이스4[Item+Category+Author] 0.027467
                 케이스5[Item+Category+Publisher] 0.026361
                  케이스6[Item+Author+Publisher] 0.025097
케이스7[Item+Category+Author+Publisher(Full)] 0.023600
                       케이스1[Item+Category] 0.023053
                     케이스3[Item+Publisher] 0.021744
                        케이스2[Item+Author] 0.021647
```

- 7가지 조합의 NDCG@10 비교 챠트



- 실험 1과 비교 그래프



[분석]

- Addition vs Concatenation 방식 비교
  - Concatenation 방식이 전체적으로 약 8.8% 더 우수
    - Concatenation 평균 NDCG@10 = 0.024997
    - Addition 평균 NDCG@10 = 0.022974
  - 피처 수가 적을 때 (2~3개) Concatenation이 대부분 우세
    - 피처 간 정보 독립성이 유지되어, 모델이 각 피처의 latent 표현을 더 잘 학습할 수 있음.
  - 피처 수가 많을 때 (Full 조합, Case7) Addition이 우세
    - 다수 피처를 단순 연결(concat)하면 차원이 급격히 커지고 학습 효율이 떨어짐.
    - Addition 방식은 차원을 유지한 채 여러 피처의 정보를 효율적으로 융합하므로, 통합된 표현 학습에 강점

- 조합 패턴 결과 실험 1과 비교
  - 최적 조합패턴이 Addition 방식은 케이스7[Item+Category+Author+Publisher(Full)]로 NDCG@10이 0.027004, Concatenation가 방식은 케이스4[Item+Category+Author] 0.027467로 서로 다름
  - Concatenation 방식은 각 피처가 별도 차원 공간에 투영되므로, 모델은 각 피처의 독립적인 패턴을 학습할 수 있음
  - Concatenation 방식은 너무 많은 피처를 넣으면 차원이 과도하게 커지고 학습 난이도 증가, 과적합 위험이 생김, 적정 수준의 피처 조합 (Case4: Item+Category+Author)이 가장 효율적
    - Category와 Author는 서로 보완적이지만, Publisher를 추가하면 잡음(정보 중복) 증가 → 성능 하락

- 최적 조합 선정 근거
  - 최적 조합 : 케이스4[Item+Category+Author] NDCG@10이 0.027467(가장 높은 성능)
    - Category는 사용자의 큰 취향 방향을 잡아줌
    - Author: 세밀한 선호 차이를 구체화
    - Item: 개별 경험 데이터로 구체적 보정
  - 실제 독서/콘텐츠 소비 패턴에서도 사용자는 '분야+작가 중심의 선택'을 하는 경우가 많으며, 출판사(브랜드)는 상대적으로 결정적 요인이 아님

03. 실험 3 - Addition 방식 최적화

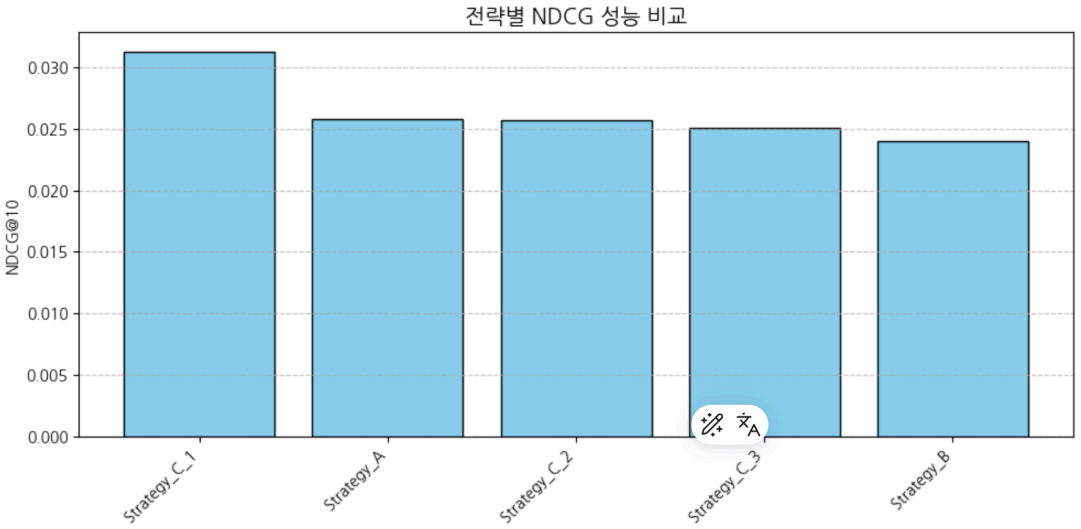
[실험 결과]

- Strategy_A : 단순 합(Sum)
    - enriched = item + Category + Author + Publisher
- Strategy_B : 평균(Average)
    - enriched = (item + Category + Author + Publisher) / 4.0
- Strategy_C : 가중 합(Weighted Sum) - 3가지 이상 시도
    - Strategy_C_1 : enriched = 1.0 * item + 2.0 * Category + 2.0 * Author + 2.0 * Publisher
    - Strategy_C_2 : enriched = 1.0 * item + 2.0 * Category + 1.0 * Author + 0.5 * Publisher
    - Strategy_C_3 : enriched = 1.0 * item + 2.0 * Category + 2.0 * Author + 3.0 * Publisher

- Sum/Average/Weighted 전략별 NDCG@10 비교 표

```
============================================================
📊 전략별 성능 비교표 (내림차순 정렬)
============================================================
    STRATEGY  NDCG@10
Strategy_C_1 0.031245
  Strategy_A 0.025795
Strategy_C_2 0.025685
Strategy_C_3 0.025070
  Strategy_B 0.023975
```

- Sum/Average/Weighted 전략별 NDCG@10 비교 챠트



[분석]

- 최적 전략 및 가중치
  - 최적 전략은 Strategy_C : 가중 합(Weighted Sum) 방식
    - Strategy_C_1인 1.0 * item + 2.0 * Category + 2.0 * Author + 2.0 * Publisher 방식이 최적의 조합

- 가중치가 성능에 미치는 영향
  - 단순 합 vs 평균
    - 단순 합은 모든 임베딩의 스케일과 중요도를 그대로 반영함
    - 평균은 각 임베딩의 기여도를 희석시켜 버림
      - Author, Publisher 같이 정보량이 많은 피처의 영향력이 줄어듦
      - 결과적으로 NDCG가 낮게 나타남
  - 가중합의 효과
    - 가중합은 각 피처의 정보 중요도·신뢰도에 맞게 조정 가능
    - Strategy_C_1의 효과적 조합
      - Category, Author, Publisher에 동일 가중치(2.0)를 주고, Item(1.0)보다 두 배로 강조.
      - 결과적으로 NDCG@10이 0.031245로 최고치 → Sum 대비 +21.1%, Average 대비 +30.3% 상승
      - 이는 Side Info(Category/Author/Publisher)가 단순 Item latent보다 정보적 가치가 크며, 적절히 강조될 때 추천 품질을 극대화하는 효과가 나타남

- 도메인 지식과의 연관성
  - Category의 가중치 2.0 : 사용자의 장르 취향은 지속적이고 예측력이 강함
  - Author의 가중치 2.0 : 작가 기반 충성도 반영 (특정 저자 선호 → 일관된 추천 성공률 상승).
  - Publisher의 가중치 2.0 : 출판사는 책의 품질·스타일 신호. 특정 출판사의 편집 방향을 신뢰하는 사용자 집단 존재.
  - Item의 가중치 1.0 : 개별 아이템 정보는 사용자의 과거 선택을 그대로 반영하지만, 새로운 상황이나 다른 아이템으로 일반화하기는 어려움

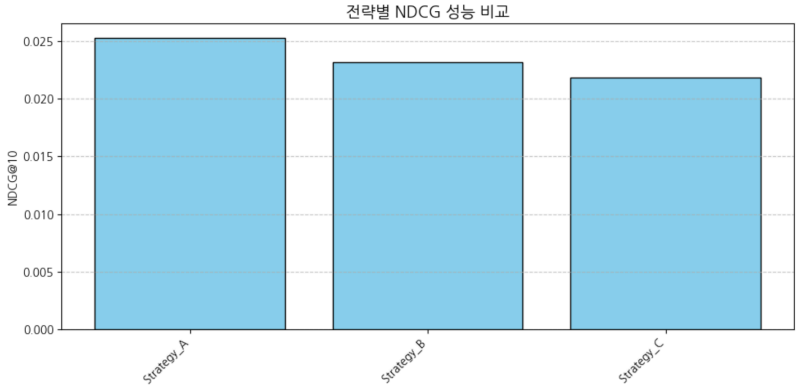
04. 실험 4 - Concatenation 차원 전략

[실험 결과]

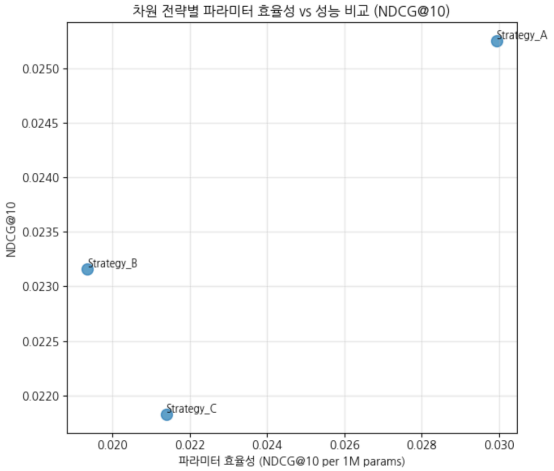
- Strategy_A : 동일 차원(Uniform)
    - Item : 8차원, Category : 8차원, Author : 8차원
- Strategy_B : 차등 차원1(자유 설계) - 중요도 기반
    - Item : 16차원, Category: 8차원, Author : 12차원
- Strategy_C : 차등 차원2(자유 설계) - 균형 조정
    - Item : 12차원, Category : 10차원, Author: 10차원
- 차원 전략별 NDCG@10 + 파라미터 수 비교 표

```
============================================================
📊 전략별 성능 비교표 (내림차순 정렬)
============================================================
  STRATEGY  NDCG@10  PARAMS  PARAMS_M  EFFICIENCY
Strategy_A 0.025250  843529  0.843529    0.029934
Strategy_B 0.023160 1197385  1.197385    0.019342
Strategy_C 0.021828 1020633  1.020633    0.021387
```

- 차원 전략별 NDCG@10 비교 챠트



- 파라미터 효율성 vs 성능 그래프



[분석]

- 차등 차원의 효과
  - 전략A는 모든 임베딩 차원을 균등(8, 8, 8) 으로 설정한 단순 구조
    - 전략B와 C는 특정 feature(예: Item, Category 등)에 더 많은 차원을 할당한 차등 전략
  - 관찰 결과
    - 차등 임베딩(전략B, C)은 특정 feature에 집중된 학습을 가능하게 하지만, feature 간 차원의 불균형이 학습 안정성에 영향을 줄 수 있음.
    - 전략B는 Item 차원을 과도하게 확대한 반면, Category 차원이 상대적으로 낮아 모델의 표현 균형이 깨졌을 가능성이 있음.

- 파라미터-성능 trade-off
  - 파라미터가 늘어나도 성능(NDCG@10)은 유사하나 효율(Efficiency)은 A → C → B 순으로 지속적으로 하락
    - 전략A – “성능, 효율 극대화 구간”
      - 가장 적은 파라미터로 준수한 성능 확보.
      - 학습 및 추론 효율이 높아 경량형 또는 실시간 추천 환경에 적합.
    - 전략B, C – “비효율 구간”
      - 파라미터 수는 많지만, 성능은 오히려 하락.
      - 특정 feature에 차원이 과도하게 집중되어 모델 표현 불균형 및 과적합 가능성.
      - 추가 파라미터가 유효 정보보다 잡음을 더 학습했을 가능성 존재

- 최적 차원 할당 전략
  - 효율성과 성능을 모두 고려할 때, 전략A (8,8,8)가 가장 합리적
  - 가장 적은 파라미터로 거의 동일한 성능 달성
    - 즉, 모든 feature를 크게 늘리기보다 중요 feature만 부분적으로 확장하는 방식이 최적.

05. 종합 분석 및 결론

[Addition vs Concatenation 최종 비교]

구분	Addition 방식	Concatenation 방식
결합 구조	피처 간 벡터를 동일 공간에서 더함(합산) → 정보 융합 중심	피처 벡터를 단순 연결(concat) → 정보 분리 중심
평균 NDCG@10	0.022974	0.024997 (+8.8%)
피처 수 적을 때	성능 낮음 (정보 손실 가능)	우수함 (특징 간 독립성 확보)
피처 수 많을 때	우수함 (정보 융합에 강함)	과적합 위험 (차원 급증)
최적 조합	Case7 [Item+Category+Author+Publisher] (0.027004)	Case4 [Item+Category+Author] (0.027467)
해석 요약	다수 피처의 상호작용을 활용할 때 강점	중복 없는 핵심 피처만 결합할 때 강점

[4가지 실험의 주요 발견]

실험 구분	핵심 발견
실험 1 – Addition 조합 분석	Full 조합(Item+Category+Author+Publisher) 시 단일모델 대비 +13.5% 성능 향상. 조합효과(Combination Effect) 명확히 존재.
실험 2 – Concatenation 조합 분석	평균적으로 Addition보다 +8.8% 우세, 특히 Item+Category+Author 조합이 최적 조합(0.027467).
실험 3 – Addition 방식 최적화 (가중합 전략)	Weighted Sum(1:2:2:2) 조합 시 최고 NDCG@10 = 0.031245. Side Info 중요도를 반영한 가중치 조정이 효과적.
실험 4 – Concatenation 차원 전략	차원 불균형 시 과적합 및 성능 저하 발생. 효율·성능 균형 측면에서 균등 차원(8,8,8) 구조가 가장 합리적.

[최종 권장 모델 (Addition 최적 vs Concatenation 최적)]

구분	추천 모델	구성	NDCG@10	특징
Addition 방식 기준 최적 모델	Weighted Sum (Strategy_C_1)	1.0×Item + 2.0×Category + 2.0×Author + 2.0×Publisher	0.031245	전체 피처 통합 + 가중 기반 정보 강화
Concatenation 방식 기준 최적 모델	Case4	Item + Category + Author	0.027467	중복 최소화 + 보완적 피처 결합
효율성 기준 최적 모델	Concatenation 차원 균등형	Item 8, Category 8, Author 8	0.025250 / 효율성 최고(0.029934)	경량형 추천 시스템에 적합

[실무 적용 시 권장 사항]

- 피처 조합 최적화
  - Full feature 조합보다는 중복 없는 핵심 피처 중심(Item+Category+Author)을 우선 고려.
- 가중치 학습(Weighted Sum) 적용
  - 도메인 지식을 반영한 피처 중요도(예: Category=2.0, Author=2.0, Publisher=2.0, Item=1.0) 활용.
- 모델 효율성 고려
  - 학습 및 추론 속도가 중요한 경우 Concatenation 균등형(8,8,8) 사용.
  - 정확도가 중요한 경우 Addition Weighted Sum(C_1) 적용.

[최종 결론 요약]

- 모델 구조와 결합 방식에 따라 성능 향상 패턴이 상이함을 확인함
- 본 실험을 통해 NCF 기반 추천모델에 Side Information(Category, Author, Publisher)을 확장 적용할 경우,
  - Concatenation은 핵심 피처 조합(Item+Category+Author)에서 최고 효율을 달성했고,
  - Addition은 가중합(Weighted Sum) 기반 Full 조합에서 최고 정확도를 보임
- 실무에서는 추천 품질 중심 → Addition(Weighted Sum), 운영 효율 중심 → Concatenation(균등 차원) 전략을 병행 적용하는 것이 가장 실용적이고 안정적인 접근으로 판단됨