당근 썸머테크인턴 포트폴리오

2024.08.11 ~ 2024.11.11

진행 프로젝트

1) 랭킹모델 학습시간 줄이기

2) 최신 연구 구현 및 실험

Software Engineer, Machine Learning 인턴 장병우

나의 강점

스스로 생각한 **본인의 강점 두 가지**

1. **학습 효율**
2. **끈기**

=> 두 가지 모두를 보여줄 수 있는 경험을 쌓자

=> **목표**: 프로젝트를 **두 개** 이상 **완수**하고, **모두 배포**하기

들어가며

세 달을 어떻게 **효율적**으로 쓸 수 있을까?

목표: **프로젝트**를 **두 개** 이상 **완수**하고, **모두 배포**하기

==========

(prj#n, nw) start_date ~ end_date: 수행한 업무 ; 핵심 성공 액션

=> 어떤 project를, n주에 걸쳐서, 어떤 업무를 수행했으며, 어떤 핵심적인 액션을 성공시켰는가

==========

(prj#1, 1w) 08.11 ~ 08.17: 학습 환경 구축, 코드 탐색 ; **학습 파이프라인** 돌리기

(prj#1, 1w) 08.18 ~ 08.24: 학습 데이터 분석 ; **데이터 샘플링 후 파이프라인** 돌리기

(prj#1, 4w) 08.25 ~ 09.26: 실험을 위한 코드 구조 파악 ; 데이터 샘플링, 모델 경량화, **다양한 오프라인 실험**

(prj#1, 2w) 09.27 ~ 10.13: 실험 플랫폼 파악 ; **온라인 실험 켜기** 및 결과 확인

(prj#1, 1w) 10.14 ~ 10.17: 배포 프로세스 파악 ; 배포 논의 및 **배포**

(prj#2, 1w) 10.18 ~ 10.27: 새로운 연구 탐색 ; MaskNet, DCNv3 **모델 구현 & 오프라인 실험**

1) 랭킹모델 학습시간 줄이기

Background

1. 당근마켓 홈 피드 아이템 노출 수 증가에 따른 데이터 포인트 양의 증가와,
2. 홈피드 랭킹모델에 여러가지 feature, 구조가 추가됨에 따라

현재의 랭킹 모델은 **학습에 이틀 넘게 걸려**, 모델 성능 향상을 위한 여러가지 실험을 해보기엔 **불편할 수 있음**

랭킹 모델 실험이 용이한 환경을 만들기 위해 **랭킹 모델 학습에 걸리는 시간 단축** 프로젝트

1) 랭킹모델 학습시간 줄이기

여러 관점의 실험 모두 진행했지만, 최종적으로 **모델 사이즈 관점 실험**에서 가장 큰 개선이 있었기에 해당 내용만 작성

1. 데이터 샘플링 관점 실험

   현재 홈피드 랭킹모델 학습에 이용되는 session 수는 2억개에 달하며, 각 session의 click, watch, chat, hide, report 기록은 최대 50개씩 존재.

   => 전체 데이터 포인트는 100억개로, 전체 데이터를 대변할 수 있는 core set을 찾기

2. 모델 Forward 관점 실험

   모델 Forward를 개선하는 실험으로, lstm과 같이 sequential model을 더욱 효율적인 model로 수정하기

3. 모델 Backward 관점 실험

   현재 홈피드 랭킹모델은 multi objective로 학습되는데, 각 objective의 weight을 수정함으로써 더욱 빠른 수렴 달성하기

4. **모델 사이즈 관점 실험**

   홈피드 랭킹모델의 layer별 파라미터 수를 파악하고, 불필요하게 큰 layer는 감축함으로써 연산속도를 높이기

1) 랭킹모델 학습시간 줄이기

레이어 별 파라미터 수

모델 사이즈 관점 실험

모델 파라미터 제일 많은 layer: keyword_vocabulary_embedding (38.4M) > scoring Model (9.54M)

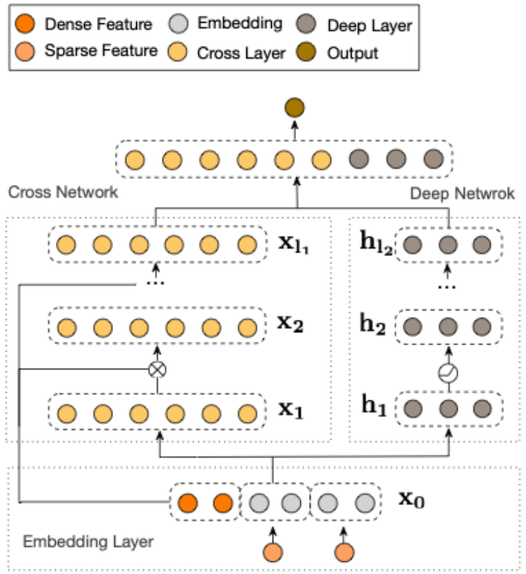
Embedding Layer	Sequential Model	Scoring Model	Total
6.4 + 38.4 + 1 + 3.2	0.35 + 0.35 + 0.35 + 0.35 + 0.35	9.54	64.17

Keyword Vocabulary Embedding 사이즈를 줄이고 batch size를 늘리는 방식을 고려해볼 수 있지만,

Vocab embedding의 경우 spare update가 일어나기 때문에, 줄인다고 해서 gpu memory가 드라마틱하게 줄어들지 않음

=> Scoring Model에 집중!

Scoring Model



(b) Parallel

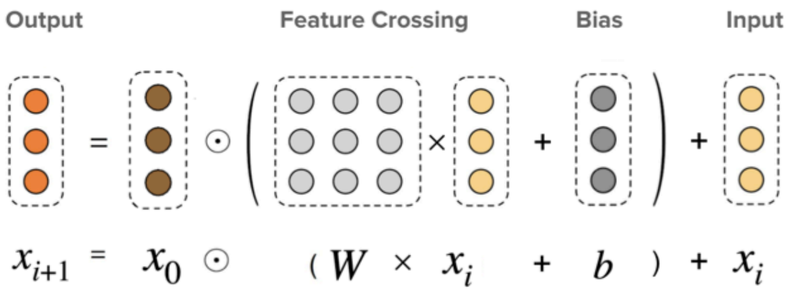


Figure 2: Visualization of a cross layer.

Scoring Model로 사용하고 있는 아키텍쳐는

2021년 구글에서 발표한 DCN v2라는 모델

여기서 Cross Network는 **input dimension이 d**인 embedding

이 들어올 때, **(d, d) shape의 파라미터**를 가지는 layer

왼쪽 아래사진처럼, feature와 feature를 곱하는 식으로, 어떤

feature 조합을 모델에 명시적으로 넣어줌으로써 성능 향상을 도모

(이를 feature crossing이라고 함)

즉, input embedding의 dimension이 커질 수록

파라미터 수는 제곱만큼 커짐

1) 랭킹모델 학습시간 줄이기

Scoring Model에 들어가는 Embedding 크기를 효율적으로 압축하자!

Embedding Layer의 파라미터 수는 **전체 모델 파라미터 수의 58%**를 차지할 정도로 많이 존재

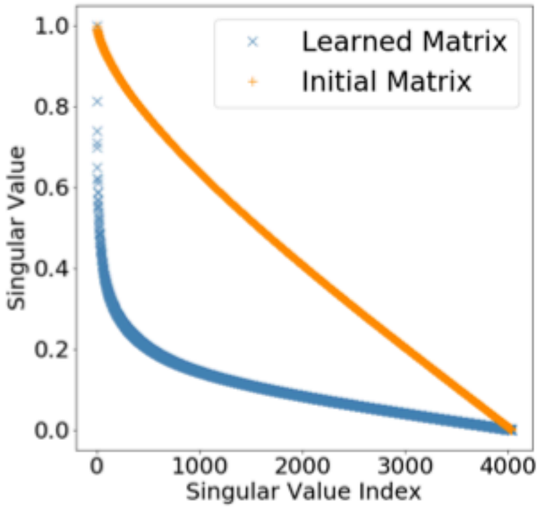
input embedding의 dimension을 줄인다면, Embedding Layer의 수도 줄이고, Scoring Layer의 연산량도 줄일 수 있음

=> 이후 슬라이드에서 소개하는 **두 가지 키포인트**를 바탕으로 **모델 경량화에 성공**

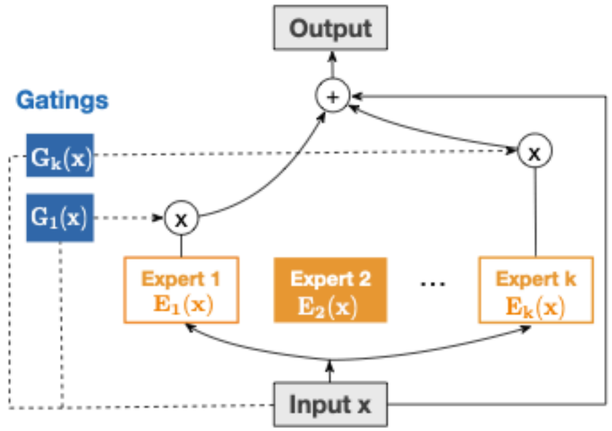
키포인트 1) Low Rank

이를 저자들도 인지하고, Cross Network의 rank를 확인해본 결과 weight matrix의 rank가 낮음을 파악할 수 있었음

(이 점을 이용하여 Weight Factorization 기법을 이용해서, 계산을 압축하는 방법을 논문에서 소개)



(a) Singular Values



(b) Mixture of Low-rank Experts

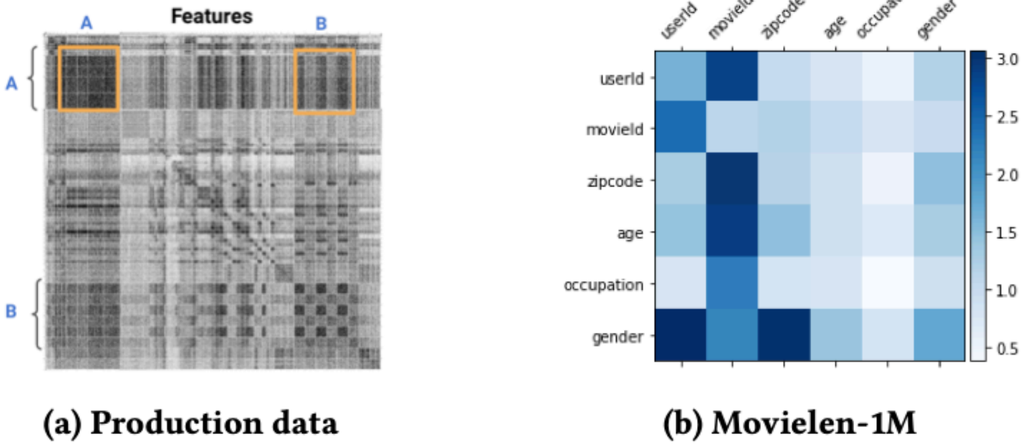
발상)

=> Cross Layer에 들어가 Feature Crossing 되는 것 중 의미 없는게 많구나

=> Cross Layer에 들어가는 임베딩을 압축하면 더 효율적이겠구나

키포인트 2) Model Interpretability

논문 후반부에, cross layer의 weight matrix를 통해 모델이 어떤 feature cross에 집중했는지를 확인할 수 있는 방법에 대한 소개



(a) Production data (b) Movielen-1M

Figure 6: Visualization of learned weight matrix in DCN-V2. Rows and columns represents real features. For (a), feature names were not shown for proprietary reasons; darker pixel represents larger weight in its absolute value. For (b), each block represents the Frobenius norm of each matrix block.

발상)

=> 어떤 feature 조합이 유효했는지 판단할 수 있구나

1) 랭킹모델 학습시간 줄이기

해결방안

1. Cross Layer에 들어가 Feature Crossing 되는 것 중 의미 없는게 많음
2. Cross Layer에 들어가는 임베딩을 압축하면 더 효율적임
3. 어떤 feature 조합이 유효했는지 판단할 수 있음

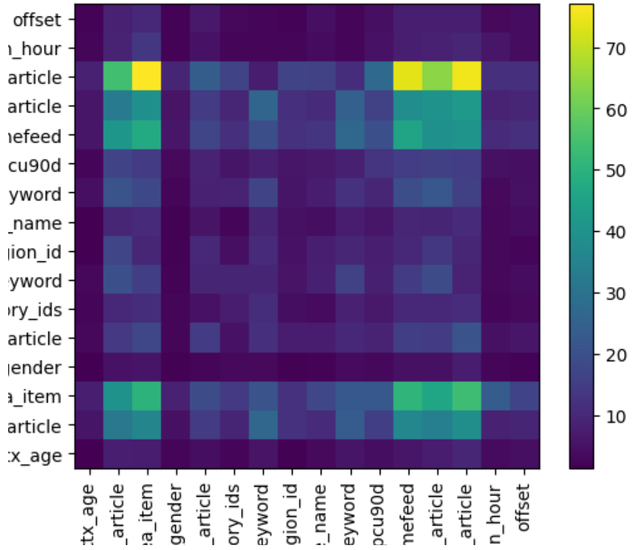
=> Cross Layer를 보면서 embedding size를 줄이고 늘리는 식으로 모델 경량화 진행

(이때, catboost나 permutation importance 같은 방법을 사용하지 않고 Cross Layer를 이용한 이유는

실제 서빙되고 있는 모델이 판단하는 피쳐의 중요도를 파악하기 위함)

1) 랭킹모델 학습시간 줄이기

Consideration



DCN v2에서 소개한 모델 해석 방법은

1. weight matrix를 Feature 단위로 나누기 (gender, age, …)

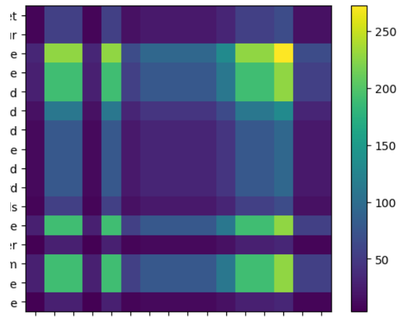
2. block의 크기 별 Frobenius norm 구하기 $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$

그렇게 되면 feature 조합의 절대적 영향력에 대해 확인할 수 있음

하지만 **알고 싶었던 것은 embedding 차원 수 대비 feature 조합의 영향력**

$$|A| = \sqrt{\frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

Frobenius norm으로만 판단하면, 임베딩 차원 크기가 단순히 클수록 높은 값을 지님

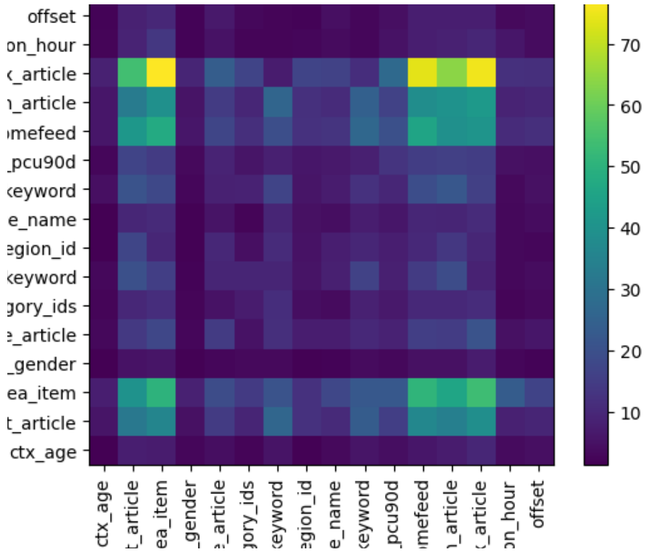


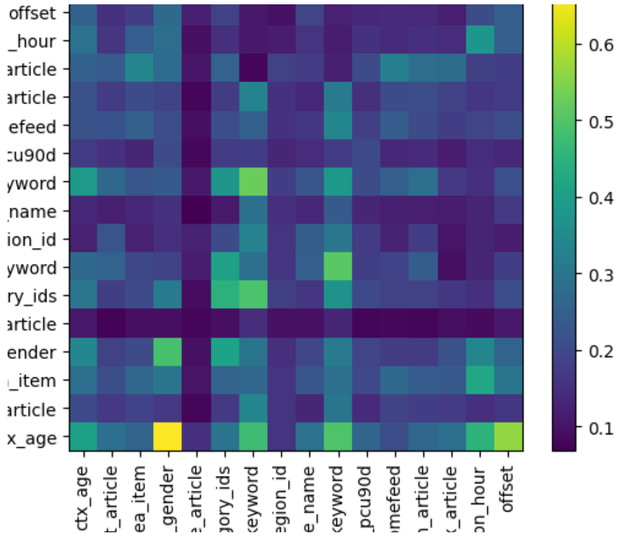
모든 값이 1일 때의 예시

(모든 차원이 동일하게 기여했지만, 차원이 큰 피쳐의 크기가 큰 것으로 나옴)

1) 랭킹모델 학습시간 줄이기

Consideration





(수정 전) $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$

(수정 후) $|A| = \sqrt{\frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$

주요 인사이트

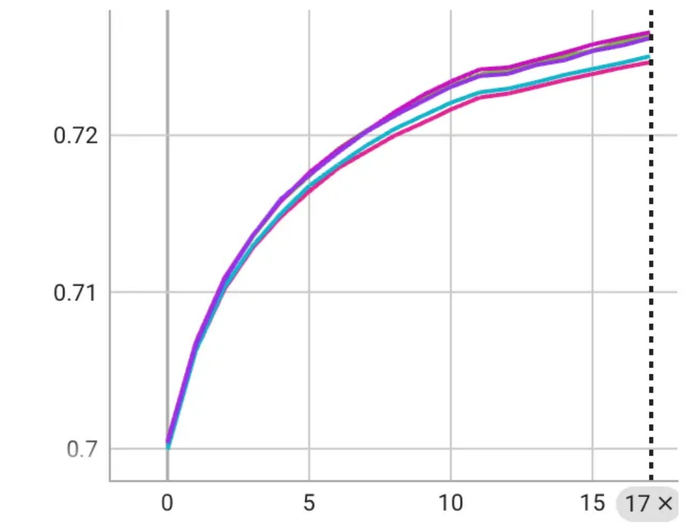
1. gender는 사이즈가 작아 수정 전에는 필요없어 보이지만, 수정 후엔 임베딩 차원대비 중요도가 낮지 않음
2. ctx_hide_article은 차원 수가 작지 않은 편이므로, 수정 후에 유독 임베딩 차원대비 중요도가 낮음으로 나옴

1) 랭킹모델 학습시간 줄이기

오프라인 실험

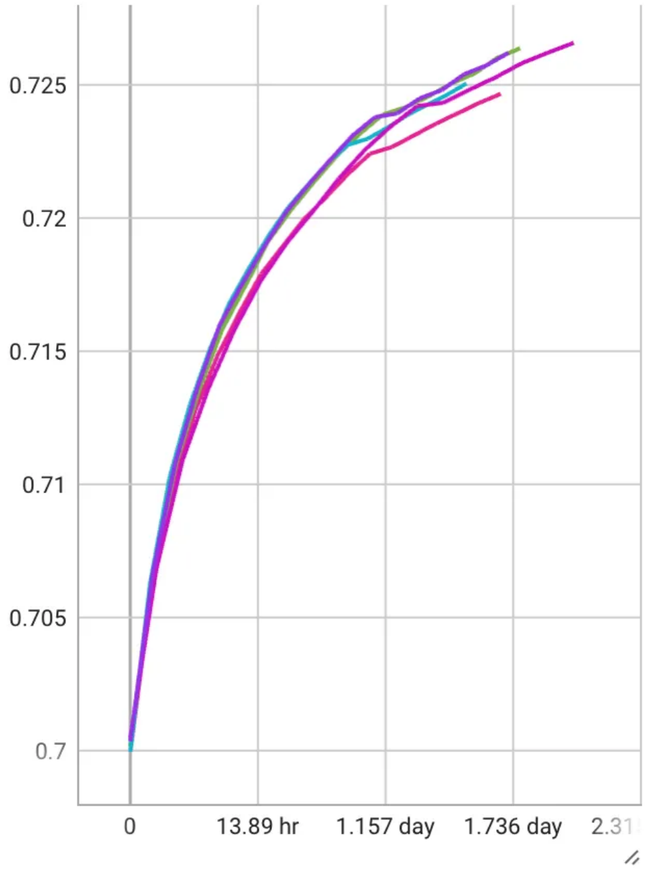
여러 feature의 사이즈를 변화시키며 오프라인 실험 감행

epoch_article_clicked_OPA

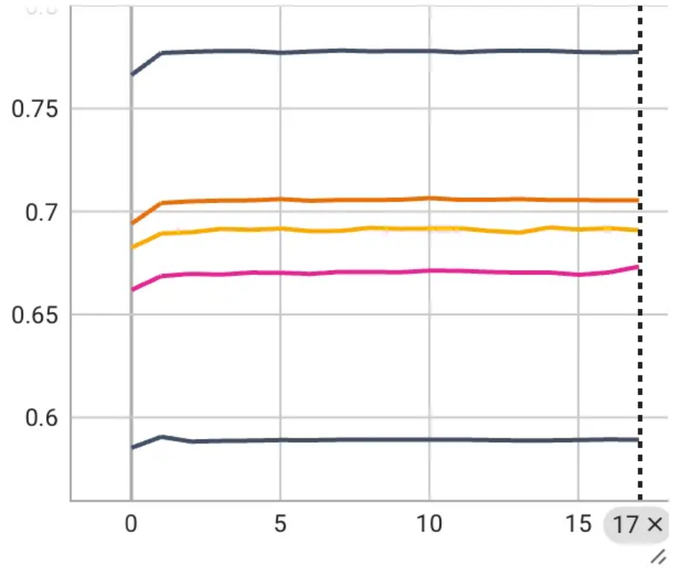


Run	Value	Step
baseline/tensorboard/validation	0.7266	17
less20/tensorboard/validation	0.7262	17
less30/tensorboard/validation	0.725	17
less30_prj/tensorboard/validation	0.7264	17
less40_prj/tensorboard/validation	0.7247	17

epoch_article_clicked_OPA



epoch_steps_per_second



Run	Value	Step	R
baseline/tensorboard/train	0.5891	17	2.
less20/tensorboard/train	0.6909	17	1.
less30/tensorboard/train	0.7775	17	1.
less30_prj/tensorboard/train	0.6732	17	1.
less40_prj/tensorboard/train	0.7054	17	1.

1) 랭킹모델 학습시간 줄이기

온라인 실험

온라인 실험 결과 (10/4 ~ 10/13, 10일치 결과)

실제 학습에 걸린 시간 (10월 10일 pipeline, Trainer 기준)

	control	t1	t2	t3
started	Oct 11, 2024, 3:46:17 AM	Oct 11, 2024, 3:19:05 AM	Oct 11, 2024, 3:43:04 AM	Oct 11, 2024, 3:43:19 AM
completed	Oct 13, 2024, 7:07:56 AM	Oct 12, 2024, 11:44:16 AM	Oct 12, 2024, 6:30:06 PM	Oct 12, 2024, 4:01:26 PM
elapsed	51 (2days 3hr)	32 (1day 8hr)	38 (1day 14hr)	36 (1day 12hr)

1) 랭킹모델 학습시간 줄이기

온라인 실험

차이 있음 지표만 가져왔으며, t2의 경우 시간 향상은 30%, 성능 하락은 없었으므로 t2 배포!

▼ 실험결과 (control vs t1)

지표 이름	지표 설명	날짜	그룹 A	그룹 B	분석 결과	P Value	A 그룹 실험 모수	B 그룹 실험 모수	A 그룹 평균	B 그룹 평균	평균값 상대 차이
clicked_fleamark	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0.005	507,838	507,684	1.134	1.101	-2.90%
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0.001	507,869	507,703	4.973	4.874	-2.00%
converted_feed_	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0.039	507,862	507,701	3.261	3.212	-1.50%
impressed_item_	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0.044	507,884	507,719	930.675	940.144	1.00%
impressed_flean	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0.026	507,883	507,714	828.171	837.716	1.20%
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0	507,900	507,739	63.066	64.158	1.70%
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0	507,900	507,739	53.883	55.021	2.10%
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment1	차이 있음	0	507,900	507,739	29.742	30.648	3.00%

▼ 실험결과 (control vs t2)

지표 이름	지표 설명	날짜	그룹 A	그룹 B	분석 결과	P Value	A 그룹 실험 모수	B 그룹 실험 모수	A 그룹 평균	B 그룹 평균	평균값 상대 차이	신뢰구간 (lower)	신뢰구간 (upper)
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment2	차이 있음	0.024	507,869	507,499	4.973	4.908	-1.30%	-0.121	-0.009
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment2	차이 있음	0.039	507,900	507,534	63.066	63.334	0.40%	0.013	0.522
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment2	차이 있음	0.002	507,900	507,534	53.883	54.231	0.60%	0.131	0.565
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment2	차이 있음	0	507,900	507,534	29.742	30.188	1.50%	0.309	0.584

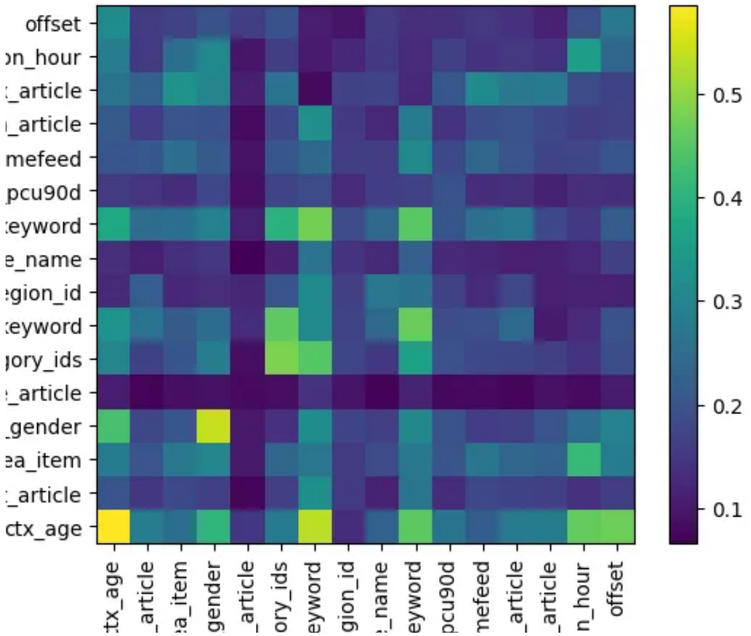
▼ 실험결과 (control vs t3)

지표 이름	지표 설명	날짜	그룹 A	그룹 B	분석 결과	P Value	A 그룹 실험 모수	B 그룹 실험 모수	A 그룹 평균	B 그룹 평균	평균값 상대 차이	신뢰구간 (lower)	신뢰구간 (upper)
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,869	506,382	4.973	4.793	-3.60%	-0.236	-0.124
click_home_feec	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,878	506,388	0.182	0.176	-3.50%	-0.01	-0.003
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,874	506,387	18.103	17.549	-3.10%	-0.802	-0.306
clicked_home_fe	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,862	506,389	1.356	1.315	-3.00%	-0.063	-0.019
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,870	506,392	39.779	38.75	-2.60%	-1.537	-0.521
impressed_hom	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0.001	507,867	506,390	10.589	10.329	-2.50%	-0.417	-0.103
impressed_item_	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,889	506,407	111.363	108.635	-2.40%	-3.746	-1.71
click_home_feec	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0.011	507,850	506,372	2.43	2.377	-2.20%	-0.094	-0.012
impression_hom	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,875	506,397	28.267	27.652	-2.20%	-0.93	-0.3
clicked_item_in_	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,879	506,388	5.948	5.82	-2.20%	-0.191	-0.066
clicked_home_fe	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0.024	507,850	506,383	0.565	0.553	-2.10%	-0.023	-0.002
converted_feed_	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0.006	507,862	506,376	3.261	3.196	-2.00%	-0.112	-0.018
watch_count	1인당 중고거래	2024-10-13	control	treatment3	차이 있음	0.042	507,845	506,357	2.147	2.105	-2.00%	-0.083	-0.002
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,900	506,417	63.066	63.614	0.90%	0.292	0.803
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,900	506,417	53.883	54.534	1.20%	0.433	0.869
unique_impressc	1인당 홈 피드	2024-10-13	control	treatment3	차이 있음	0	507,900	506,417	29.742	30.326	2.00%	0.447	0.723

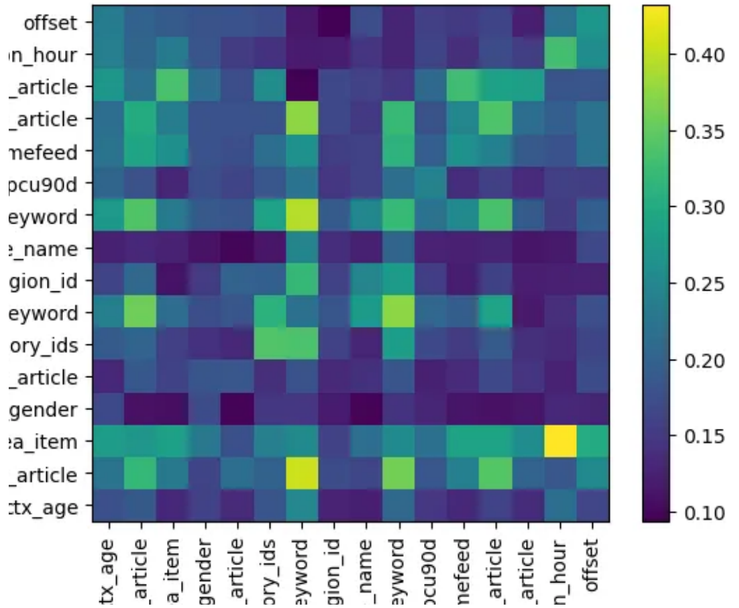
1) 랭킹모델 학습시간 줄이기

전후 cross layer 비교

피쳐 차원 대비 중요도가 고르게 펴진 것을 확인할 수 있음



(Before)



(After)

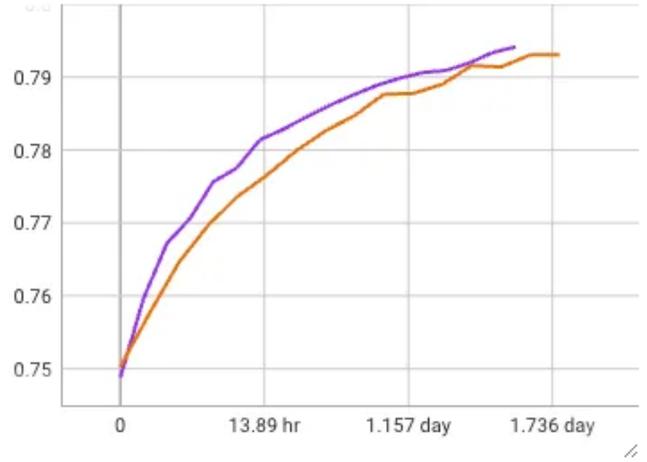
2) 최신 연구 구현 및 실험

MaskNet, DCN v3

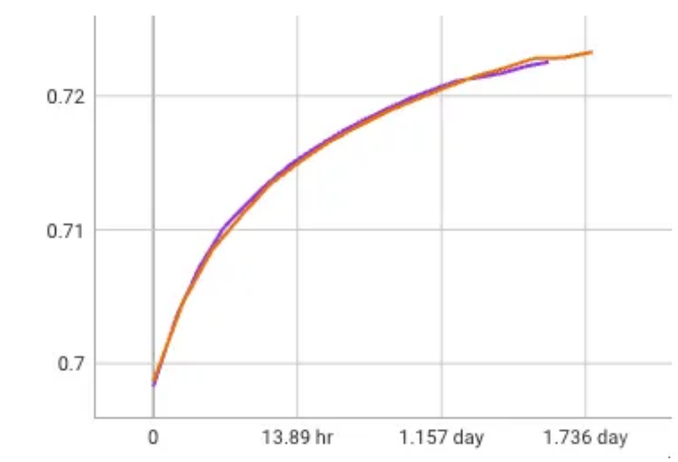
2) 최신 연구 구현 및 실험

MaskNet, DCN v3

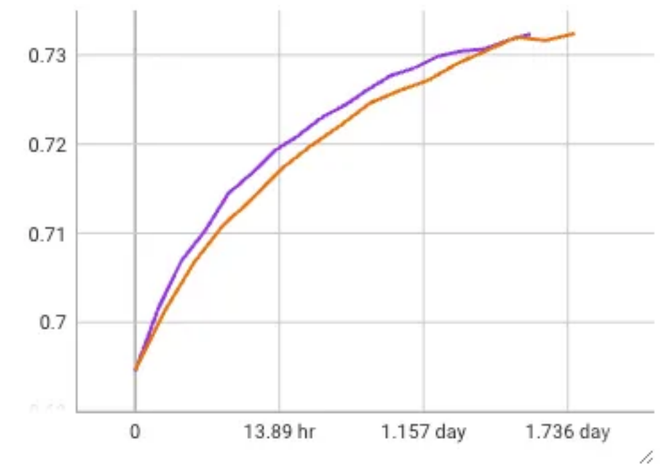
epoch_article_chatted_OPA



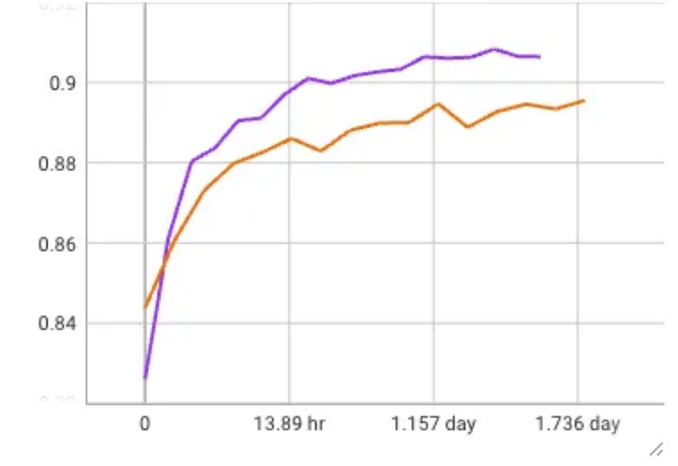
epoch_article_clicked_OPA



epoch_article_watched_OPA



epoch_article_reported_OPA



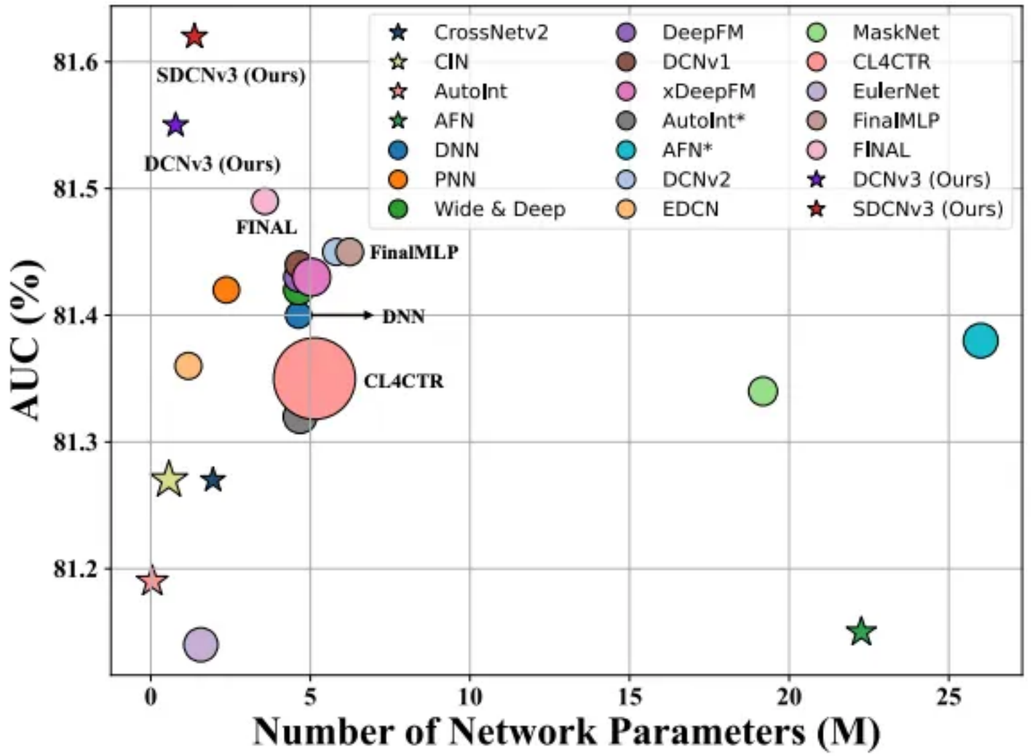
Masknet (보라)과 baseline (노랑)의 성능 비교

Masknet이 시간 대비 성능 향상 폭이 높음

현재 리뷰 중

2) 최신 연구 구현 및 실험

MaskNet, DCN v3



DCN v3의 경우 파라미터 수 대비 성능이 가장 좋음

그러나 실제 구현 시, 파라미터 수는 실제로 작으나 연산속도가 느린 현상 확인

해당 현상 디버깅 진행 중