Embedding-based Recommendation for Millions of Users

Property	
Property 1	
Property 2	
abstract	뉴스 추천에서 기사 내용이나 유저 Preference를 이해하는 것은 필수적이다. ID 기반이나 low-rank factorization은 잘 알려진 추천 방법이나 뉴스 추천 시스템에는 적절하지 않다. 그 이유는 후보 기사들이 빨리 만료되고 새로운 기사로 교체 되기 때문이다. word 기반 방법은, 검색에서는 시스템 성능 면에서는 좋을 수가 있지만, 문제가 있는데 동의어와 맞춤법 등에 대처가 필요하고 유저 이에서 Query 정의가 필요하다. 논문에서는 세 단계 End-To-End 방식으로 분산된 표현을 사용하는 Embedding 기반 방법을 소개한다. (1) Denoising AutoEncoder로 분산된 기사를 처리하는 것부터 시작해서 (2) 유저 Input Sequence을 사용해서 RNN으로 유저 표현을 생성하고 (3) 유저와 제품의 내적을 통해 기사들을 추천하는데 시스템의 성능을 고려했다 Yahoo Japan의 데이터를 오프라인 테스트해서 성능을 보였다. 온라인 테스

	트로 CTR을 봐서 23%의 성능 개선을 보고 Duration을 10% 개선했다. Yahoo Japan은 천만 유저 서비스이다.
기타	Scalability : 환경에 따라 Application이 확장 될 수도 축소 될 수도 있는 능력
논문 링크	file:///Users/kangseokwoo/Downloads/Embedding-based%20Recommendation%20for%20Millions%20of%20Users.pdf
발제자	강석우
발표자	석우 강
키워드	News Recommendation, Neural Networks, Distributed Representation, Large-Scale Service
한글 링크	

Introduction

시간 제약이 있기 때문에 유저가 수 많은 뉴스를 볼 수는 없다. 그래서 유저는 선택적으로 공급되는 뉴스들을 보는 것을 더 선호한다. 선택하는 방법은 manual 하게는 TV나 신문들에서 편집자의 선택 등으로 공통적으로 공급 될 수 있다. 하지만 인터넷에서는 Cookie 등을 이용해서 개인화된 추천을 할 수 있다. CF나 MF가 추천에서 유명하지만 컨텐츠의 교체가 빠르기 때문에 뉴스 추천 에는 적절하지 않다. 뉴스 추천에서 중요한 Key 세 가지가 있다.

1. 뉴스 내용 이해
2. 유저 Preference 이해
3. 뉴스와 Preference 기반으로 추천하기

추가로 현실적인 문제로 Scale과 Data의 Noise가 있다.

세 가지를 다루기 위한 Baseline 을 구현한다면 세 Key는 다음과 같이 다루어진다. 기사들은 단어들로 구성된 텍스트. 유저도 본인이 봤던 기사들의 단어들로 텍스트. 그러면 Candidate 기사와 Browsing History의 Word Co-Occurence를 Feature로 사용해서 CTR을 학습한다. 실용적인 이점이 있다. 방법이 간단하기 때문에 짧은 기간으로 학습이 가능하다. → 짧은 주기로 업데이트를 할 수 있다. → 트렌드를 잘 반영한다. 그리고 Word Inverted Index를 가지는 검색 시스템과 연계해서 빠른 우선 순위 계산도 가능하다. Baseline 모델의 문제는 두 가지가 있다.

1. Representation of Words. 동의어의 경우에 다른 Feature로 취급이 되고, 같은 사건에 대해 같은 의미의 동의어가 사용된 여러 개의 기사가 등장 하면 문제가 생긴다.

2. Handling Browsing History. 사실상 Sequence Data고 Sequence 순서로 유저의 관심 변화를 표현해야 한다. 그리고 유저 마다 시퀀스의 길이가 다 다르고 이걸 처리 해야 한다.

단어의 Distributed Representation은 의미 정보를 반영 해야 한다. RNN이 효과적으로 길이를 가진 정보를 처리할 수 있다. RNN으로 Preference와 Content를 기반 해서 Interest를 예측은 근데 실제 시스템에서 Response Time을 맞추는 게 어렵다. 그래서 논문에서는 데이터의 중복과 관련성을 해결하는 추천 방법을 제공한다.

(1) Denoising AutoEncoder로 분산된 기사를 처리하는 것부터 시작해서
(2) 유저 Input Sequence을 사용해서 RNN으로 유저 표현을 생성하고
(3) 유저와 제품의 내적을 통해 기사들을 추천하는데 시스템의 성능을 고려했다

내적을 사용했기 때문에 응답 시간이 빠르다. 사용자가 서비스에 접근하면 해당 유저의 표현을 가져와서 후보 뉴스들에 내적으로 추천을 한다. 스마트폰 서비스에 적용 되었다. 다른 상업적인 접근 방법에 비해서 좋았다. 모델 업데이트에 대한 학습 시간 증가와 지연 시간 증가에 대한 문제가 있긴 하다.

2. 서비스와 프로세스 흐름

메소드는 Yahoo Japan의 스마트 폰의 뉴스 서비스에 적용 되었다.

Figure 1: Example of Yahoo! JAPAN's homepage on smartphones. This paper discusses methods of providing articles in Personalized module.

]다섯 개의 Process 들이 수백만 유저들의 각각 접근 들에 대한 기사 선택으로 실행되었다.

- Identify : 유저 History로 부터 유저 Feature를 획득
- Matching : 사용 가능한 유저 Feature들로 기사들을 추출
- Ranking : 정해진 우선순위로 기사들을 재 정렬

- De-Duplication : 같은 정보를 담고 있는 기사들을 제거
- Advertising : 필요하다면 광고를 삽입

이러한 프로세스 가 유저 Request 를 밀리 세컨드 단위로 수행된다. 정보가 참신해야 하기 때문에 24시간 내에 뉴스를 만료 시킨다. 수 만의 뉴스들이 새롭게 게시되고 사라진다. 그래서 각각의 프로세스가 경량화 되어야 한다. 랭킹을 위한 우선순위는 추가적인 Factor를 고려해서 결정하는데 예상되는 조회 수와 각 기사들의 참신성을 추가로 고려한다. 기사의 중복을 제거하기 위해 Cosine-Similarity를 사용해서 정해진 Threshold를 넘어가면 넘기는 Greedy Method를 적용한다.

3. Generating Method

3.1 Generating Method

De-noising Auto-encoder에 기반으로 조금 개선해서 분산된 표현으로 된 벡터를 생성한다.

$$
\begin{aligned}
\tilde{x} &\sim q(\tilde{x}|x) \\
h &= f(W\tilde{x}+b) \\
y &= f(W'h+b') \\
\theta &= \underset{W,W',b,b'}{\arg\min} \sum_{x\in X} L_R(y,x),
\end{aligned}
$$

$x \in X$는 원본 입력 벡터이고 $q(\cdot|\cdot)$ 은 노이즈가 추가된 분포다. 확률적으로 노이즈가 추가된 벡터 $\tilde{x}$는 $q(\cdot|x)$로 만들어진다. hidden representation, h 는 $\tilde{x}$로 부터 파라미터 W와 b와 함께 activation 함수로 구성되어 있는 네트워크 $f(\cdot)$를 통해 매핑된다. 재 구성되는 y는 마찬가지로 파라미터들과 함께 h로 부터 매핑된다. loss 함수 $L_R(\cdot,\cdot)$를 사용해서 y와 x의 Reconstructed Error를 최소화한다. h는 보통 representation 벡터로 x에 할당되어 사용되지만, h는 오직 x의 정보만을 가진다. Consine 유사도가 높아질 수록 유사도가 높아지도록 하기 위해

카테고리적인 유사도를 보존시킨다. Triplet, $(x_0, x_1, x_2) \in X^3$를 입력으로 사용하고, objective 함수를 변경한다.

$$
\begin{aligned}
\tilde{x}_n &\sim q(\tilde{x}_n|x_n) \\
h_n &= f(W\tilde{x}_n + b) - f(b) \\
y_n &= f(W'h_n + b') \\
L_T(h_0, h_1, h_2) &= \log(1 + \exp({h_0}^{\mathrm{T}} h_2 - {h_0}^{\mathrm{T}} h_1))
\end{aligned} \tag{1}
$$

$$
\theta = \underset{W, W', b, b'}{\arg\min} \sum_{(x_0, x_1, x_2) \in T} \sum_{n=0}^{2} L_R(y_n, x_n) + \alpha L_T(h_0, h_1, h_2),
$$

$T \in X^3$고, x_0과 x_1은 같거나 유사한 카테고리이고 x_0와 x_2는 다른 카테고리다. 수식 1의 h는 다음과 같은 특성을 만족하는데, x = 0 ⇒ h = 0. 이용 가능한 정보가 없는 글은 다른 글들과 유사하지 않다는 뜻이다. 표현식 $L_T(\cdot, \cdot, \cdot)$은 패널티 함수이다. 카테고리적인 유사도에 할당되는 기사 유사도를 위한 패널티 함수이다. α는 밸런싱을 위한 하이퍼 파라미터다.

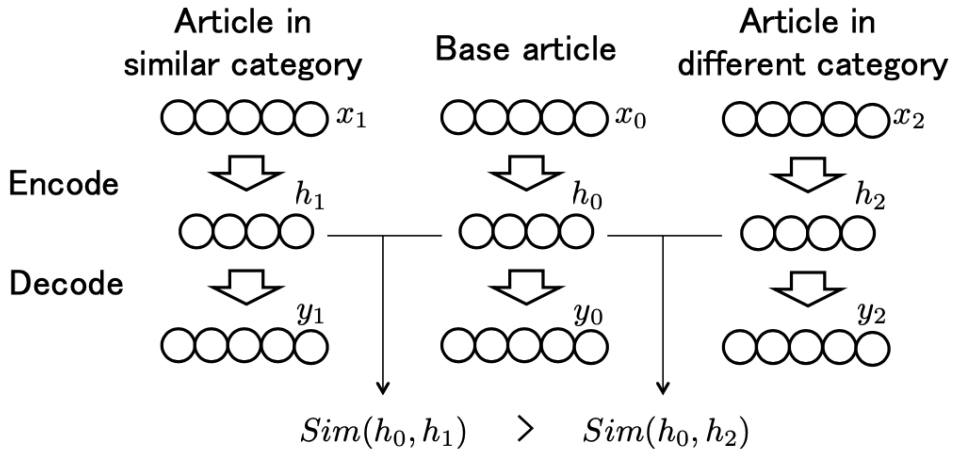

Figure 2: Encoder for triplets of articles

element-wise sigmoid 함수 $f(\cdot)$를 $\sigma(x)_i = 1/(1 + exp(-x_i))$로 사용하고, element-wise cross-entropy는 $L_R(\cdot|\cdot)$, 그리고 노이즈를 만드는데 $q(\cdot|\cdot)$를 사용한다. 모델으로 파라미터 W, b를 학습한다. mini-batch SGD를 사용한다.

$\tilde{x}$를 application phase에서 constant decay를 사용해서 생성하고, training phase에서 확률적인 노이즈를 만들지는 않는다.

$$\begin{aligned}\tilde{x} &= (1-p)x \\ h &= f(W\tilde{x}+b) - f(b),\end{aligned}$$

p는 training phase에서 corruption phase이다. 그래서 h는 application of time에서 unique 하게 결정된다. $1-p$를 곱하는 것은 노이즈 마스킹과 이런 노이즈 없이 학습하는 것 사이에서 중간층의 각 뉴런에 대해 입력 분포를 균등화 하는 효과가 있다.

위에서 생성한 h는 세 개의 application에서 기사를 Representation 할 때 사용한다.

1. 섹션 4에서 user state 함수에 입력한다.
2. matching 에서 유저와 기사의 관련성을 측정할 때 사용한다.
3. 중복 제거에서 기사들 간의 similarity를 측정할 때 사용한다.

4. User Presentations

4.1 Notation

A는 기사들의 전체 셋이다. Representation of element는 $a \in A$의 method에 대해 의존한다. a는 Word-Based 메소드의 sparse vector다. 벡터의 각 요소는 Vocabulary의 각 단어에 해당된다. 그러나 a는 기사의 distributed representation vector다.

Browse는 기사들의 페이지 URL에 접근을 의미한다. $a_t^u \in A_{t=1,\dots,T_u}$ 는 유저 $u \in U$의 browsing history라고 한다.

Session은 유저가 추천 서비스에 방문하고 추천 리스트 내에서 기사를 하나라도 클릭하는 것을 의미한다.

유저 u가 추천 리스트에서 뉴스를 클릭하면 (세션이 발생한다) 그/그녀 는 해당 URL로 접근을 하게 되고 (브라우징이 발생한다) 그래서 Session과 Browse a_t^u와 a_{t+1}^u는 하나 이상의 Session이 발생할 수 없다. 그러므로 Session은 s_t^u라고 할 수 있다. 그러나 유저 u는 서비스가 없이도 기사 URL에 접근할 수 있다. 웹 검색을 사용해서 접근할 수 있다. 그래서 s_t^u가 항상 있는 건 아니다.

세션이 사용자 u에게 할당되는 추천 리스트 이기 때문에, 세션을 s_t^u로 표현하고 $\{s_{t,p}^u \in A\}_{p \in P}$ 리스트 형태이다. Notation $P \subseteq N$ 은 이러한 Session 내에서의 추천 리스트 에서의 위치 집합이다. $P_+ \subseteq P$ 는 클릭 된 위치이고 $P_- = P \setminus P_+$는 클릭 되지 않은 위치 들이다. P, P_+, P_-는 유저 u와 브라우징 t에 달려 있지만, 표기를 간단하게 하기 위해서 첨자를 생략한다.

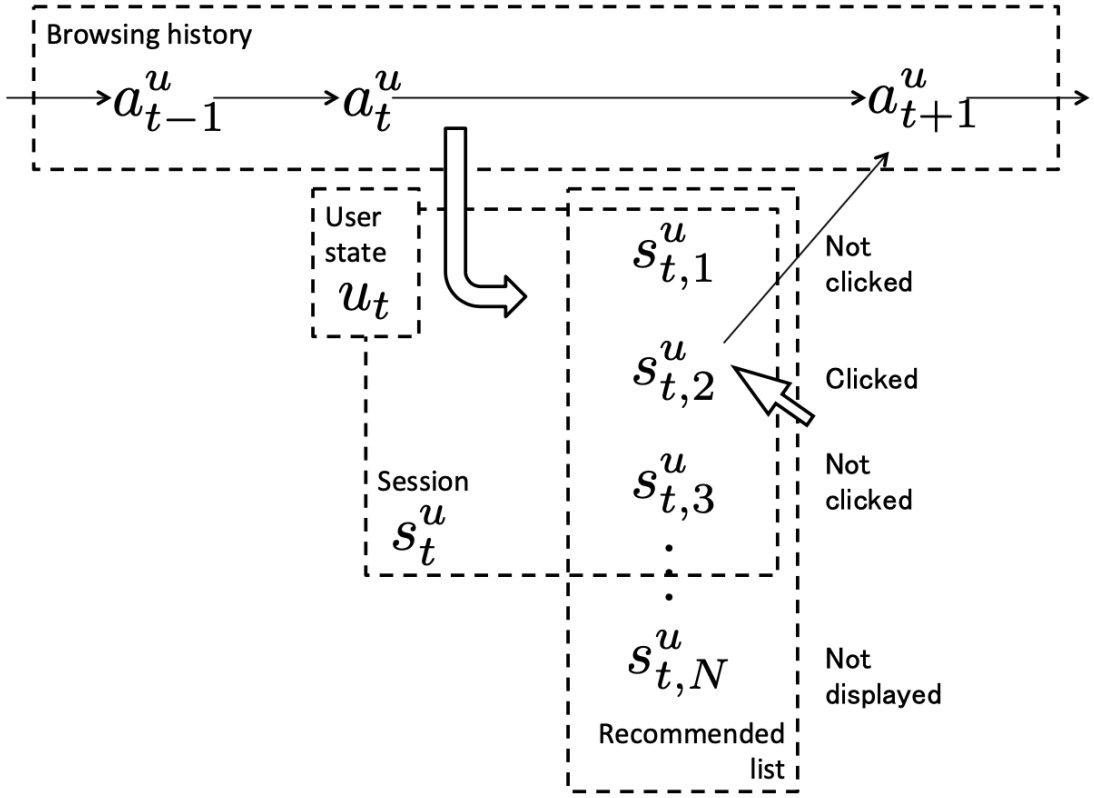

Figure 3: Browsing history and session

u_t를 user state라고 한다. $a_1^u ... a_t^u$ 에 의해서 그래서 u_t는 a_t^u라는 브라우징 후에 유저 u의 preference라고 나타낸다. $R(u_t, a)$는 user state의 u_t와 뉴스 a의 관련성이다. 시간 t에서의 뉴스 a에 대한 유저 u의 선호의 정도. 본 연구에서의 주요 목적은 user-state function F를 만드는 것이고 관련성을 측정하는 함수 R을 만드는 것이다.

$$u_t = F(a_1^u, \ldots, a_t^u)$$

$$\forall s_t^u \, \forall p_+ \in P_+ \, \forall p_- \in P_- \, R(u_t, s_{t,p_+}^u) > R(u_t, s_{t,p_-}^u). \qquad (2)$$

실제로 많은 트래픽의 뉴스 분포 시스템에서의 제약된 반응 시간에 맞춰야 하기 때문에, 함수 R은 빨리 계산되는 간단한 함수 여야 한다. 후보 뉴스들이 빠르게 교체되기 때문에 모든 유저들 $\{u_t | u \in U\}$ 에 대해서 $R(u_t, a)$에 해당하는 관련도 점수를 계산 할 수 없기 때문이다.

관련성을 계산하는 함수 R을 간단한 내적인 $R(u_t, a) = u_t^T a$로 제한하고 미리 계산할 수 있는 user-state 함수 F만 최적화 한다.

$$\sum_{s_t^u} \sum_{\substack{p_+ \in P_+ \\ p_- \in P_-}} -\frac{\log(\sigma(R(u_t, s^u_{t,p_+}) - R(u_t, s^u_{t,p_-})))}{|P_+||P_-|}, \tag{3}$$

σ는 logistic sigmoid 함수이다. 수식 4.1은 수식 2를 펼친 것이다. 노출되는 위치에 따라 편향성이 있기 때문에, 실제 상황에서는, 목적 함수에 B()를 넣어서 편향성을 바로 잡는다. 편향성을 바로 잡는 B()는 실제 학습되는 parameter 지만, B()는 다음에 많이 나올 것이라서 수식 에서는 생략 되었다.

$$\sum_{s_t^u} \sum_{\substack{p_+ \in P_+ \\ p_- \in P_-}} -\frac{\log(\sigma(R(u_t, s^u_{t,p_+}) - R(u_t, s^u_{t,p_-}) + B(p_+, p_-)))}{|P_+||P_-|}.$$

4.2 Word-Based Model

이전에 논문에서 언급 되었던 Baseline 모델을 구현 단계로 서술한 것이다.

$$\begin{aligned}
&V : \text{Vocabulary set} \\
&a, u_t \in \{0,1\}^V \\
&(a)_v = \begin{cases} 1 & (\text{if the article contains the word } v) \\ 0 & (\text{otherwise}) \end{cases} \\
&(u_t)_v = (F(a_1^u, \ldots, a_t^u))_v = \alpha_v \max_{1 \le t' \le t} (a^u_{t'})_v,
\end{aligned} \tag{4}$$

$(x)_v$는 x의 v 번째 원소이다. 관련성 함수는 parameter로 $\{\alpha_v\}$를 가지는 간단한 선형 모델이다.

$$\begin{aligned}
R(u_t, a) &= u_t^{\mathrm{T}} a \\
&= \sum_{v \in V} (u_t)_v (a)_v \\
&= \sum_{v \in V} \alpha_v 1_{v \in u_t \cap a}.
\end{aligned}$$

모델은 두가지 Issue가 있다.

1. Representation of Words. 동의어의 경우에 다른 Feature로 취급이 되고, 같은 사건에 대해 같은 의미의 동의어가 사용된 여러 개의 기사가 등장 하면 문제가 생긴다.
2. Handling Browsing History. 사실상 Sequence Data고 Sequence 순서로 유저의 관심 변화를 표현해야 한다. 그리고 유저 마다 시퀀스의 길이가 다 다르고 이걸 처리 해야 한다.

어떻게 Issue를 본 논문의 방법으로 해결 했는지를 이제 알려준다

4.3 Decaying Model

1. Bag Of Words 방식을 사용하지 않았으므로, 1번은 쉽게 해결 되었다.
2. Browsing History를 집계하는데 최대 값을 사용하지 않고, Weighted Average를 사용해서 집계한다. 최근 히스토리에 좀 더 가중치를 부여한다.

$$u_t = \alpha \odot \frac{1}{\sum_{1 \le t' \le t} \beta^{t-t'}} \sum_{1 \le t' \le t} \beta^{t-t'} a^u_{t'},$$

α는 a^u_t와 동일한 차원을 가지는 parameter 벡터 고, $\odot$는 두 벡터의 element-wise 곱, $0 < \beta \le 1$는 스칼라 값으로 time decay의 강도를 나타내는 hyper-parameter다. β가 1이면, 집계는 간단한 평균이 되고, 브라우징 순서는 고려되지 않는다. Baseline 모델과 유사하게 모델에서는 α만 학습되면 된다.

4.4 Recurrent Models

4.4.1 Simple Recurrent Unit

decaying model에서의 word-based model에서의 문제를 어느 정도 해결 했지만, 여전히 exponential decaying을 했을 때의 Frequency를 선형적으로 처리하는 것과 Forget 문제는 여전히 남아 있다.

좀 더 일반화를 하자면, u_t는 이전 유저 상태 u_{t-1}와 이전 브라우징 a^u_t로 결정 되어야 한다.

$$u_t = f(a^u_t, u_{t-1}).$$

그래서 RNN을 사용해서 학습을 해야 한다. 간단한 RNN 공식은 다음과 같다.

$$u_t = \phi(W^{in} a_t^u + W^{out} u_{t-1} + b),$$

ϕ는 활성화 함수 $tanh()$다. 초기에 사용 되는 u_0는 u에 의존하지 않고 모두 같은 값으로 사용한다. mini-batch SGD를 사용해서 학습하고 4.1에서의 목적 함수를 사용한다. RNN의 자체적인 한계인 Gradient Vanishing/Exploding 문제는 여전히 남아 있다. 다른 구조적으로 추가된 두가지 모델을 사용할 수 있다.

4.4.2 Long-Short Term Memory

$$\begin{aligned}
gi_t &= \sigma(W_{gi}^{in} a_t^u + W_{gi}^{out} u_{t-1} + W_{gi}^{mem} h_{t-1}^u + b_{gi}) \\
gf_t &= \sigma(W_{gf}^{in} a_t^u + W_{gf}^{out} u_{t-1} + W_{gf}^{mem} h_{t-1}^u + b_{gf}) \\
enc_t &= \phi(W_{enc}^{in} a_t^u + W_{enc}^{out} u_{t-1} + b_{enc}) && (5) \\
h_t^u &= gi_t \odot enc_t + gf_t \odot h_{t-1}^u && (6) \\
go_t &= \sigma(W_{go}^{in} a_t^u + W_{go}^{out} u_{t-1} + W_{go}^{mem} h_t^u + b_{go}) \\
dec_t &= \phi(W_{dec}^{mem} h_t^u + b_{dec}) && (7) \\
u_t &= go_t \odot dec_t, && (8)
\end{aligned}$$

Center Flow가 Main로 입력(browsed 기사)에서 시작해서 출력(user state)으로 Flow한다. 입력 a_t^u는 뉴스 벡터 공간으로 부터 hidden space로 인코딩 된 것(수식 5)이고 이전 hidden state과 병합되서(수식 6), 뉴스 벡터 공간으로 user state로 사용되도록 디코딩 된다(수식 7, 8).

Input Gate는 유저 상태를 구성하기 위해 불필요한 입력을 걸러 낸다.(뜬금없는 관심이 원인이 된 입력)

Forget Gate는 유저에 의한 흥미의 감소를 나타낸다. Decaying Model에서 사용 했던 exponential decay 보다 더 복잡한 Forget 효과를 나타낼 수 있다.

Output Gate는 다음 세션 에서 Focusing 하지 않아야 하는 것 들을 걸러 낸다.

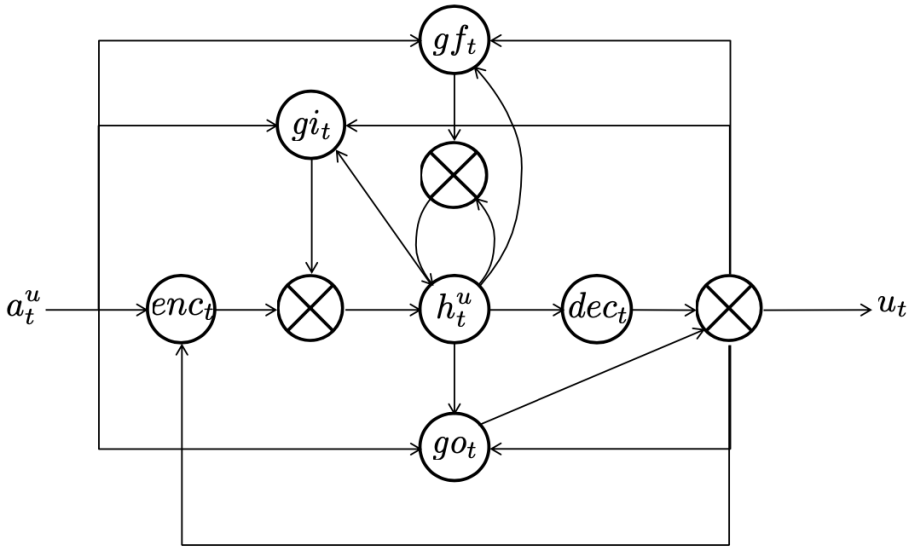

Figure 4: LSTM-based model

4.4.3 Gated Recurrent Unit

$$
\begin{aligned}
gz_t &= \sigma(W_{gz}^{in} a_t^u + W_{gz}^{mem} h^{t-1} + b_{gz}) \\
gr_t &= \sigma(W_{gr}^{in} a_t^u + W_{gr}^{mem} h^{t-1} + b_{gr}) \\
enc_t &= \phi(W_{enc}^{in} a_t^u + W_{enc}^{out}(gr_t \odot h^{t-1}) + b_{enc}) \\
h_t^u &= gz_t \odot enc_t + (1 - gz_t) \odot h_{t-1}^u \qquad (9) \\
dec_t &= \phi(W_{dec}^{mem} h_t^u + b_{dec}) \qquad (10) \\
u_t &= dec_t .
\end{aligned}
$$

정확하게 말하자면 모델은 한 개의 GRU 레이어를 사용하고 한 개의 Fully-Connected 레이어를 사용한다. 수식 10은 기존 GRU에는 존재하지 않는다. LSTM과 유사하지만 차이가 있는데. 수식 6과 9 사이에서 LSTM과 차이를 보인다. 이 모델에서 gz_t 게이트는 두 게이트의 역할을 한다. LSTM에서 gi_t, gz_f의 역할을 한다. norm의 상한 $||h_t^u||_\infty$에서 차이를 보인다.

$$
\sup_u ||h_t^u||_\infty = \begin{cases} ||h_0^u||_\infty + t \sup_x |\phi(x)| & \text{in LSTM} \qquad (11) \\ \max(||h_0^u||_\infty, \sup_x |\phi(x)|) & \text{in GRU .} \qquad (12) \end{cases}
$$

수식 11은 시퀀스 길이가 엄청나게 길다면 큰 값이 될 수 있다. 그러나 수식 12는 커져도 상수 넘지는 못한다. 그래서 GRU가 LSTM에 비해 Gradient Exploding 문제에 더 적합하다고 저자는 생각했어요. GRU는 뭘 안 해도 Exploding이 안되는데 LSTM은 Exploding 때문에 뭔가를 더 해야 한다.

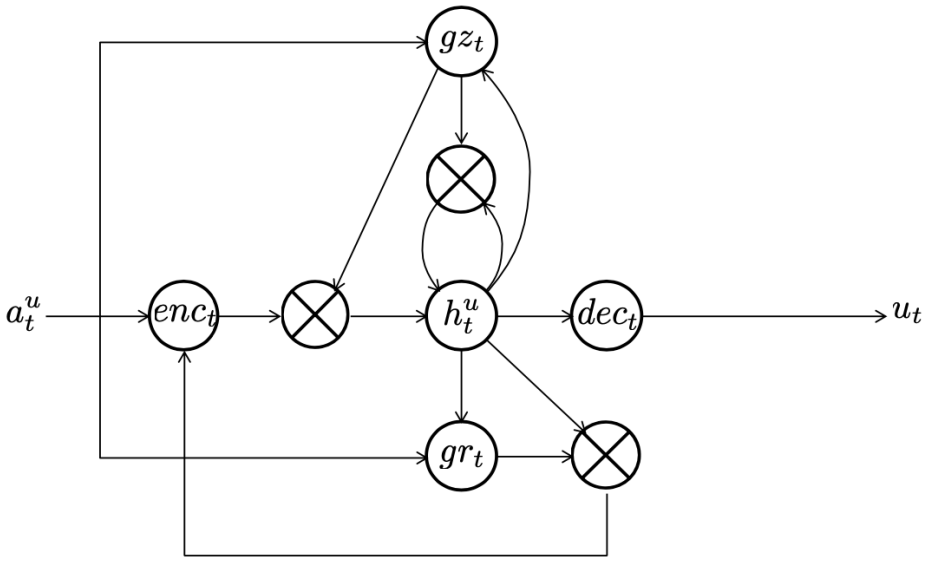

Figure 5: GRU-based model

5. Offline Experiments

과거 서빙 로그로 3개의 Word-Based 모델과 5개의 Distrubted Representatin-Based 모델들을 비교한다.

5.3 Offline Metrics

랭킹을 ROC 커브(AUC), MRR(Mean Reciprocal Rank), nDCG(normalized Discounted Cumulative Gain)를 사용해서 Evaluation 한다.

S는 세션들의 집합

$c_{s,i}$는 Position i에서 뉴스가 클릭 됐을 때, 1 이다.아니면 0 이다.

π는 위치의 임의 순열이다.

$$
\begin{aligned}
\text{AUC} &= \frac{1}{|S|}\sum_{s\in S}\frac{|\{(i,j)|i<j, c_{s,i}=1, c_{s,j}=0\}|}{|\{i|c_{s,i}=1\}||\{j|c_{s,j}=0\}|} \\
\text{MRR} &= \frac{1}{|S|}\sum_{s\in S}\frac{1}{\min\limits_{c_{s,i}=1} i} \\
\text{nDCG} &= \frac{1}{|S|}\sum_{s\in S}\frac{\sum_i c_{s,i}/\log_2(i+1)}{\max_\pi \sum_i c_{s,\pi(i)}/\log_2(\pi(i)+1)},
\end{aligned}
$$

Table 3: Average metric lift rates in 7th week and those by user segments.

Metric	ALL	Heavy	Medium	Light
Sessions	+2.3%	+1.1%	+1.0%	+1.8%
Duration	+7.8%	+4.9%	+13.3%	+17.4%
Clicks	+19.1%	+14.3%	+26.3%	+42.3%
CTR	+23.0%	+18.7%	+29.8%	+45.1%

6.3 Experimental Results

방문 빈도에 따라서 세그먼트를 세 개로 정의 했다. $Heavy : Medium : Light = 3 : 2 : 1$ 다.

- Heavy : 전 주에 5일 이상 방문
- Medium : 전 주에 2-5일 방문
- Light : 전 주에 2일 이하 방문

Table 4: Average daily percentage of user composition in 7th week for both buckets.

Bucket	ALL (%)	Heavy (%)	Medium (%)	Light (%)
Control	100.0	50.0	33.9	16.1
Proposed	100.0	51.4	33.6	15.0
0th week (ref.)	100.0	49.7	34.3	16.0

6.4 Deployment challenges to large-scale deep-learning-based services

세 시간 마다 최신 뉴스 데이터와 세션 데이터를 모델을 업데이트 하기 위해 사용한다. GRU 기반의 모델은 그러나 네가지 이유로 자주 업데이트 할 수 없다.

- Main Traffic에 사용 되는 모델은 GPU를 사용해도 일주일이 걸린다
- 모든 이용 가능한 뉴스들에 대해 Representation을 재 계산해야 하고 뉴스 검색 엔진에 re-index 해야 한다
- 유저 Representation을 업데이트 하면, user state를 처음부터 새로 로드 해야하고 옛날 히스토리는 다 버릴 수 밖에 없다
- 유저 Representation과 뉴스 Representation은 동시에 업데이트 되야 한다

그래서 두 동일한 시스템을 사용해서 이 주마다 한번 씩 모델을 업데이트 한다. 경험적으로 Word-Based Model은 유행어에 민감해서 방치하면 몇 일이 지나면 성능이 감소하지만 이 주마다 업데이트로 성능을 유지 할 수 있다. GRU 모델을 3개월 동안 방치하면 3시간 마다 업데이트하는 Word-Based Model의 정확도로 성능이 악화된다.

7. Conclusion

Embedding 기반의 방법을 3단계로 구성된 End-To-End 방식을 본 논문에서 소개한다. 많은 트래픽이 발생하는 시스템 이라서 개인화된 추천을 하기 위해 내적을 사용했다. 현재 Yahoo Japan의 스마트폰 홈페이지에서 매일 수백만 유저에게 서비스를 하고 있다. 광고 같은 도메인에 적용하려고 하고 있다.