Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 4, pp. 463-472, August 2020
https://doi.org/10.34163/jkits.2020.15.4.002

**Journal of Knowledge Information Technology and Systems**

ISSN 1975-7700

http://www.kkits.or.kr

# A Dynamic Recommendation Architecture and Procedure Based on Elasticsearch

**Ji-Hyun Jeong, Chul-Jin Kim***

*Department of Computer Systems and Engineering, Inha Technical College*

## ABSTRACT

E-commerce product recommendation uses various recommendation techniques. The recommendation techniques that are generally applied include the collaborative filtering technique, the user based collaborative filtering, and the item based collaborative filtering technique, and utilizes recommended techniques based on artificial intelligence technologies such as Recurrent Netural Network (RNN) and Long Short Term Memory (LSTM). The recommendation data generated through these recommendation techniques are stored in a database and are recommended in the form of static data by user actions (search, order, etc.). However, the recommendation through the static recommendation product list is limited in improving the purchasing power of the user. Providing a variety of recommended products and real-time properties, it is possible to further improve purchasing power if it is possible to dynamically construct a recommended product list. Therefore, this paper proposes an architecture and procedure for providing dynamic recommendations. The proposed dynamic recommendation architecture constructs a dynamic recommendation product by applying a static recommendation product based on Elasticsearch. In the experiment, accuracy was compared and analyzed by applying the same transaction data to existing recommendation techniques, and it was confirmed that the accuracy of the dynamic recommendation technique proposed in this paper is improved compared to the existing recommendation techniques.



**KEYWORDS:** Dynamic recommendation, Static recommendation, Elasticsearch, Recommendation architecture, Accuracy

**ARTICLE INFO:** Received 15 June 2020, Revised 6 July 2020, Accepted 10 August 2020.

*Corresponding author is with the Department of *Computer Systems and Engineering, Inha Technical College,* 100 Inha-ro Nam-gu Incheon, 22212, KOREA.
*E-mail address*: cjkim@inhatc.ac.kr

## 1. 서 론

온라인 쇼핑몰에서 사용자가 높은 관심을 보이는 상품을 적절하게 추천하는 것은 매출액의 상승과 직결된다[1]. 이에 따라 온라인 쇼핑몰 상에서 사용자의 다양한 관심도를 기반으로 관심 있는 상품을 분석하여 예측하는 협업적 필터링 기법[2], 클릭과 같은 상호작용을 활용하는 세션(Session) 기반 추천 시스템[3] 등의 다양한 추천 기법이 적용되었다. 그러나 이러한 기법들은 각종 상호작용으로 발생한 정보들에 대해 실시간으로 재학습을 시키기에는 어려움이 존재하며, 특히 세션 기반 추천 시스템 같은 경우에는 학습에 의해 만들어진 후보군들에 대해 새로운 상호작용 데이터로 재학습이 되기 전까지는 정적인 데이터라는 한계가 있다.

본 연구에서는 학습으로 만들어진 정적인 추천 후보 데이터를 통해 추천하는 방법에 국한되지 않고, 정적 데이터와 랭킹 함수를 활용하여 사용자의 관심 상품을 지속적이고, 동적으로 추천할 수 있는 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 사용되는 추천 시스템에 관한 연구에 대해 분석하며, 3장에서는 본 연구에서 제안하는 동적 추천 기법에 관한 아키텍처와 절차를 제안한다. 4장에서는 본 연구에서 제안하는 동적 추천 아키텍쳐를 기반으로 추천 모델을 개발하며, 기존 연구와 성능 비교 분석을 수행한다. 5장에서는 본 논문이 갖는 차별성을 바탕으로 결론을 제시하며, 향후 연구 방향을 제시한다.

## 2. 관련연구

본 장에서는 동적 추천의 기반이 되는 스코어링 알고리즘과 추천 시스템 관련연구로서 순환신경망 기반 추천 시스템과 세션 기반 추천 시스템을 연구한다.

### 2.1 TF-IDF 알고리즘

TF-IDF(Term Frequency-Inverse Document Frequency) 알고리즘[4]은 검색 엔진에서 사용되며, TF와 IDF의 곱으로 계산되는 형태의 알고리즘이다. 주로 단어의 등장 빈도수를 통해 문서들에 대해 스코어링(scoring)하여 문서 간 유사도를 판단한다.

$$score(q,d) = \sum_{t \in q} tf(t,d) * IDF \quad (1)$$

$$IDF = \log \frac{|D|}{df(t)+1} \quad (2)$$

식 (1)에서 $tf(t,d)$는 단어 빈도수인 TF(Term Frequency)를 나타내며, 개별 문서 $d$에 검색 단어 $t$가 등장한 횟수로 계산한다. 만일 특정 문서에 'word'라는 단어가 4번 들어갔을 경우 TF는 4가 되며, 문서 내에 단어가 자주 등장할 경우 의미 있는 단어로 계산된다.

IDF는 역문서 빈도로, 전체 문서에서 검색 단어에 대한 가중치를 조정하기 위한 식이다. 전체 문서 집합에서 자주 등장하는 단어일수록 의미가 없는 단어로 계산된다.

이와 같은 방식으로 질의에 대해 각 문서에 대한 점수를 산출하여 연관성 있는 문서를 검색할 수 있으나, 특정 문서 내에 질의에 사용된 단어의 개수가 많아질수록 TF가 발산하여 검색된 문서의 부정확성이 발생할 수 있다는 문제가 존재한다.

### 2.2 BM25 알고리즘

BM25(Best Maching 25) 알고리즘[5]은 검색 엔진에서 문서 간의 유사성을 산출해내기 위한 알고리

즘으로, score를 토대로 문서들이 얼마나 유사한지를 판단한다. 식 (3)을 통해 유사한 문서들을 score로 도출 후 점수가 높은 순, 즉 순위로서 정렬하여 나타낸다.

$$score(q,d) = \sum_{t \in q} IDF \cdot \frac{tf_{t,d}}{tf_{t,d} + k \cdot Norm} \quad (3)$$

$$IDF = \log(1 + \frac{N - df_t + 0.5}{df_t + 0.5}) \quad (4)$$

$$Norm = (1 - b + b\frac{l(d)}{avgdl}) \quad (5)$$

식 (3)은 검색 질의 $q$에 대해 특정 문서 $d$에 대한 점수를 산출하는 공식이다. 특정 문서 내에 존재하는 각 단어 $t$의 점수의 합으로 score가 계산이 되며, $tf_{t,d}$는 해당 문서에서 등장하는 단어 $t$의 빈도수, $k$는 상수값을 의미하며, TF이다. 식 (4)는 역문서 빈도로, 단어 $t$가 전체 문서에서 등장 빈도수의 역수를 조정한 식이다. 식 (5)는 TF의 점수 편중을 막기 위한 표준화이다. $q$에 대해 식 (3)을 적용하여 높은 score를 지닌 문서를 산출하며, 이는 가장 유사도가 높은 추천 리스트이다.

BM25는 표준화 작업을 통해 특정 문서의 TF가 높아질 경우 이에 비례하여 score가 높아지지 않고 특정 값에 수렴한다는 점에서 기존의 TF-IDF보다 데이터 검색에 대한 안정성을 보인다.

## 2.3 순환 신경망을 활용한 세션 기반 추천 시스템

연구 [3]은 순환 신경망을 세션 기반의 추천 시스템에 적용한 연구로서 기존 추천 시스템으로 널리 사용되던 Factor Model은 사용자들의 특정 상황에서의 정보 부재로 인해 세션 기반 추천 시스템에 활용하기 어렵다는 점에 기인하여 세션의 순차적 클릭에 대한 정보를 순환 신경망에 적용하였다. 평가를 위한 데이터셋으로 2015년 RecSys Challenge에서 제공한 클릭 세션에 대한 정보가 포함된 데이터와, Youtube와 비슷한 형태의 OTT(Over The Top Service) 플랫폼에서 형성된 일정 시간 이상의 영상을 시청한 사용자의 세션 정보를 사용하였다.

성능 비교를 위한 기준점이 되는 모델은 item-KNN(k-nearest neighbors)을 사용하였고, 실제 모델 구성은 GRU를 통해 Hidden size를 100개와 1,000개로 각각 지정한 후, rmsprop과 adagrad를 최적화 함수로 구성하였다. 또한 최종 도출된 GRU 모델의 성능을 item-KNN과 비교하였으며, Hidden size가 100이고 Adagrad를 활용하였을 때 기준 모델보다 약 20~30%의 성능 향상이 있음을 입증하였다.

연구[3]은 사용자의 순차적인 클릭 정보를 시계열로 구성하였고 이를 순환 신경망에 직접 적용함으로써 추천 시스템에 강점을 보이는 기존 KNN 기법보다 월등한 성능을 도출하였다는 점에 의의가 있다. 그러나 본 연구에서는 사용자의 클릭 정보가 아닌 구매 데이터를 활용한다는 점에서 연구[3]과 차이가 있으며, 또한 순환 신경망은 새로운 추천을 만들어내기 전까지는 정적인 추천 데이터를 기반으로 한다는 측면이 차별성을 보인다.

## 2.4 임베딩을 활용한 순환 신경망 추천 기법

연구 [6]은 연구 [7]에서 제안한 가중치 결합 기법을 통하여 입력 데이터에 대한 임베딩(Embedding)을 수행한 후, 각 데이터에 대해 유사도를 측정한 후 이를 순환 신경망에 적용한 추천 시스템 연구이다. 활용 데이터로는 2014년 RecSys

Challange 데이터와 Movielens 1M 데이터를 활용하였으며, user-KNN(k-nearest neighbors), BPR-MF(Bayesian Personalized Ranking-Matrix Factorization), FPMC(Factorizing Personalized Markov Chains), Fossile, LSTM(Long Short-Term Memory), 그리고 가중치 결합 기법을 활용한 임베딩 기법이 적용된 LSTM 등의 모델을 통해 평가를 진행하였다. 평가 방법으로는 모델 구성을 위해 사용자가 특정한 시점까지 시청한 영화의 기록으로 입력 데이터를 구성하고, 타겟 데이터로는 그 이후에 시청한 영화 데이터를 바탕으로 구성하였으며, 또한 학습 모델이 생성한 추천 리스트를 순회하며 사용자가 실제로 시청한 영화가 있을 경우 추천에 성공하였다고 판단하여 정확도를 도출하였다. 연구 [6]의 경우 기존 LSTM 기반 추천 시스템과 비교하여 약 1~2%의 유의미한 정확도를 제공한다.

연구 [6]은 임베딩 기법을 변형하여 LSTM을 학습시켜 그 정확도를 기존보다 더 높였다는 측면에서 의미가 있으나, 정적인 데이터를 기반으로 추천되며 추가적인 동적인 데이터를 추천하기 위해서는 재학습을 해야 한다.

## 3. 동적 상품 추천 아키텍처 및 절차

본 연구에서는 순환신경망 기반의 추천 시스템을 통해 생성된 정적인 추천 상품 리스트를 기반으로 재학습 없이 동적 상품을 추가하여 추천하기 위한 연구로서 동적 상품 추천 아키텍처 및 절차를 제안한다.

### 3.1 동적 추천 아키텍처

본 연구에서는 스코어링 알고리즘인 BM25를 사용하는 Elasticsearch 엔진[8]을 동적 상품 추천 아키텍처에 적용한다.

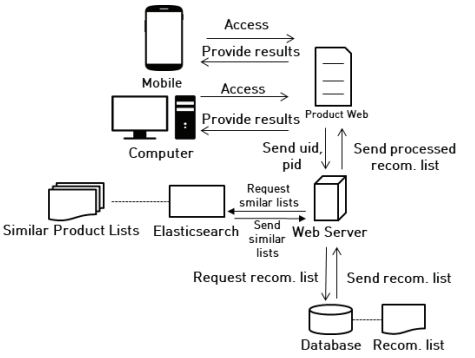



그림 1. 동적 추천 아키텍처
Figure 1. Dynamic Recommendation Architecture

본 연구에서는 두 가지의 목표를 제시한다. 첫 번째, 동적으로 검색되어 추천되는 상품을 정적인 추천 리스트에 추가하여 추천한다. 이는 정적인 추천 리스트라고 하더라도 순환신경망 기반의 추천 시스템에 의해 학습되고, 그 결과로 생성된 의미 있는 데이터라고 할 수 있다. 두 번째는 동적으로 추천되는 상품은 특정 사용자에 대해 만들어진 정적인 추천 리스트 내의 상품과 유사해야 한다. 사용자가 관심 상품에 접근 하였을 때 데이터베이스 내에 존재하는 정적 추천 상품과 동적으로 추가된 상품의 특성이 관심 상품과 연관성이 없다면 이는 경험적으로 신뢰하기 어려울 수 있다.

Elasticsearch 기반의 동적 추천을 위한 아키텍처는 <그림 1>과 같다. 사용자는 Mobile, PC와 같은 환경에서 추천 상품이 포함된 웹에 접근한다. 해당 웹은 웹서버로 접속한 사용자의 아이디인 uid와 접근한 상품의 아이디인 pid를 전송한다. 이후 웹서버는 정적 추천 리스트가 담긴 데이터베이스에 uid와 pid를 보내고, 해당 id와 일치하는 정적인 추천 리스트를 검색한 후 웹서버로 전달한다. 웹서버는 전달 받은 데이터를 다시 Elasticsearch 서버에 전송하며, 전달받은 정적 상품 리스트에 대해 순위함수를 통해 유사한 추천 리스트들을 검색한다. 그 중 score가 가장 높은 리스트를 도출하여 웹서버로

전송하고 웹서버는 정적 상품 리스트의 상품과 score가 높은 추천 상품 리스트가 지닌 상품 중에서 중복되는 추천 상품을 제거한다. 최종적으로 본래의 정적 리스트와 새로 도출된 추천 리스트를 통합하여 동적 구성된 추천 상품 리스트를 제공한다.

### 3.2 동적 추천 상품 도출 절차

본 연구에서는 동적으로 추천 상품을 관리하기 위해 Elasticsearch 엔진을 기반으로 하며, 정적 추천 상품과 동적 상품을 관리한다. Elasticsearch에서 내부적으로 동적 추천 상품 리스트를 도출하는 절차는 <그림 2>와 같다.

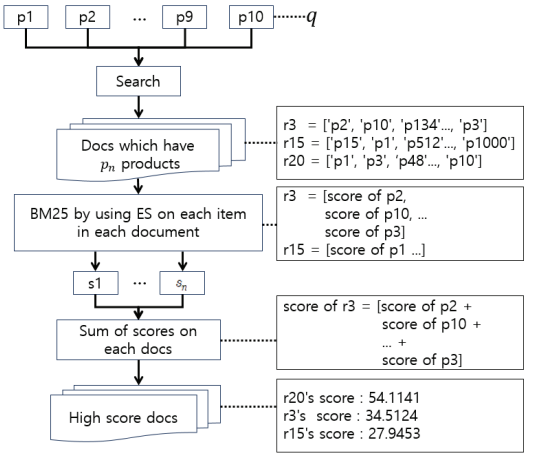



그림 2. 동적 추천 상품 리스트 도출 절차
Figure 2. Deriving Process of Dynamic Recommendation Product List

Elasticsearch는 정적 추천 리스트가 구성하고 있는 모든 개별 상품을 내부 저장소에 inverted index[9] 형태로 저장한다. 이때 웹서버로부터 전달받은 정적 추천 리스트인 $q$가 구성하고 있는 개별 상품 $p1, p2, \ldots, p10$을 Elasticsearch의 inverted index를 통해 가능성 있는 추천 리스트 $r_n$을 도출한다. 이후 BM25 연산으로 이들을 scoring 함으로써 유사 후보 추천 리스트들을 최종 도출한다. 그러나 검색에 사용될 정적 추천 리스트는 Top-10으로 항상 고정된 길이를 갖으며, 해당 추천 리스트 내에 내재된 상품은 같은 리스트 내의 상품과 중복되지 않는다. 즉, 기존의 BM25 연산을 진행할 때 TF의 값이 항상 동일한 현상이 발생하여 계산된 TF 값의 의미가 없어지는 현상이 발생한다. 이와 더불어 기존의 IDF는 전체 문서에서 자주 등장하는 단어에 대해서는 의미를 가지지 않는 단어라고 보고 가중치를 낮춰 해당 단어 대한 score를 낮게 산출한다. 이를 정적 추천 리스트의 관점에서 바라본다면 인기 있는 상품의 경우 유사 상품 리스트에 등장하는 빈도수가 다른 상품에 비해 높음에도 불구하고 의미 없는 단어로 간주되어 오히려 score가 낮게 책정되는 문제가 발생한다. 따라서 기존의 식을 약간 수정하여 계산에 사용한다.

$$score(q,d) = \sum_{t \in q} \log\left(1 + \frac{df_t + 0.5}{N - df_t + 0.5}\right) \quad (6)$$

식 (6)은 유사 후보 추천 리스트를 도출하기 위한 변형된 스코어 함수로서, 본 연구에서 사용되는 데이터의 특성에 부합하도록 기존 BM25 식을 커스터마이즈한다. 각 유사 추천 리스트 $d$의 최종 score는 검색할 정적 추천 리스트 $q$가 지닌 개별 상품 $t$에 대한 score의 합으로 산출된다. 이때, 인기 있는 상품은 전반적으로 사용자들 사이에서 선호되는 상품이므로 전체 정적 추천 리스트에서 자주 등장할수록 score를 더 높게 책정한다.

식 (6)을 통해 산출된 유사 후보 추천 리스트들은 질의에 사용된 정적 추천 리스트 내에 담긴 상품 $p_n$을 최소 1개 이상 내포하고 있다. 만일, 유사 후보 추천 리스트 $r_n$이 구성하는 상품들이 $q$과 완

전히 동일할 경우, 각 상품에 대한 score인 $s_n$의 합이 가장 높을 것이며, 한 개의 상품을 제외한 나머지 상품들만 일치할 경우, 그 다음 순위가 될 것이다. 그러나 질의에 사용된 정적 추천 리스트와 완전히 동일한 형태로 구성된 유사 후보 리스트는 사용할 이유가 없다. 결론적으로 동일한 후보이기 때문이다. 따라서, 완전히 동일한 후보군을 제외한 유사 후보군의 상품을 활용한다.

<그림 3>은 도출된 여러 유사 후보군들이 지닌 상품 중 하나가 최종적으로 도출되어 정적 추천 리스트와 병합되는 과정을 나타낸다.

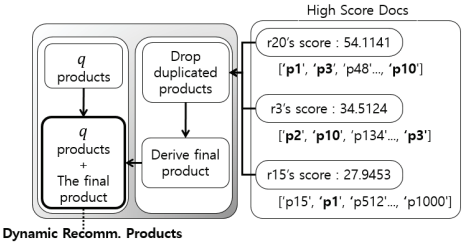



그림 3. 최종 동적 추천 리스트 도출
Figure 3. Formation of Final Dynamic Recommendation List

$r_n$이 지닌 상품들 중 기존 정적 추천 리스트 $q$ 내의 상품 $p_n$과 겹치지 않는 상품만을 골라서 최종 상품 후보 리스트를 형성한다. 이때 형성된 최종 리스트 내의 상품들 중 하나의 상품을 선택하여 기존 정적 추천 리스트와 병합하며, 선택되는 상품은 랜덤 함수를 통해 임의로 설정한다. 다만, 최종적으로 형성된 상품 후보 리스트의 상품 중, 상대적으로 score가 낮아 연관성이 떨어지는 유사 리스트의 상품도 포함하고 있으므로, 상위 10%의 점수를 가진 상품만 재취합한 후, 최종적으로 상품을 선택한다.

## 4. 실험 및 평가

본 실험에서는 제안한 동적 추천 아키텍쳐의 실효성을 증명하고, 이에 대한 정확도를 평가한다. 실험에 앞서 트랜잭션 데이터셋의 각 행으로 구성된 구매 데이터에 대해 생성된 정적 추천 리스트 86,999개에 대해 정확도를 측정한 이후 동일한 정적 추천 리스트를 동적 추천 아키텍쳐에 적용하여 최종 결과를 비교 분석한다.


그림 4. 구매 데이터에 대해 생성된 정적 추천 리스트
Figure 4. Static Recommendation List Generated for Transaction Data

<그림 4>는 기존 연구 [10] 에서 생성된 정적 추천 리스트로, 추천 모델에 의해 생성된 Top-10의 형태로 구성되어 있고, i-recom 필드의 값은 3677 고객의 3342 상품에 대한 추천 후보를 의미한다. 동적 추천 기법에 적용하기 위해 사용되는 추천 리스트 또한 동일한 형태의 데이터 이며, Elasticsearch를 기반으로 각 추천 리스트를 질의하여 동적 추천 상품을 검색한다.

```
# Elasticsearch Query Function
def es_item_search(recom_list):
    result = es.search(index='lstm_research', body={
        "size":300,
        "query":{
            "match" : {"i-recom" : recom_list}
        }
    })

    # Logic to Process Lists
    submit = {}
    i = 0
    # Print only '_source'
    for data in result['hits']['hits']:
        submit[i] = data
        i+=1

    return submit
```

그림 5. Elasticsearch 질의 함수
Figure 5. Elasticsearch Query Function

본 실험에서는 <그림 5>와 같이 동적 추천 상품을 검색하기 위해 코드를 통해 유사한 추천 리스트의 목록을 도출한다.

이때 도출된 유사 리스트는 검색에 사용된 기존 추천 리스트와 유사한 정도를 scoring하여 내림차순으로 정렬되는데, 이때 유사 리스트 중 기존 추천 리스트와 완전히 동일한 데이터를 지닌 경우 score가 가장 높게 산출된다.

```
# Extract Lists except Same Data
for j in range(len(recom_dicts)):
    if recom_dicts[j]['_score'] != recom_dicts[0]['_score']:
        temp_list.append(recom_dicts[j]['_source']['i-recom'].split(' '))
    else:
        continue
```

그림 6. 동일 데이터 제외 코드
Figure 6. Equal Data Exclusion Code

이러한 경우 결국 기존 데이터와 동일한 데이터이므로 의미가 없는 데이터로 볼 수 있다. 이를 <그림 6>의 코드를 통해 정제하여 데이터가 완전히 겹치지 않는 유사 리스트로 새롭게 구성한다.

```
[3370 3369 3380 2066 3343 2067 1610 3374 3342 3344]
[2067 3369 3343 1610 2066 3370 3380 1609 3344 3342]
[3370 2066 3343 3369 3380 2067 1610 3344 3342 3374]
[3342 3380 3370 3369 3344 2066 3343 2067 3374 1610]
```

그림 7. 유사 추천 리스트 도출
Figure 7. Derivation of Similar Recommendation Lists

<그림 7>과 같이 최종적으로 유사 추천 리스트가 도출된다. 해당 유사 추천 리스트들은 기존 추천 리스트와의 질의를 통해 도출된 결과이며, 이때 볼드체로 표시된 상품 코드(3374, 1609)는 기존 추천 리스트와 겹치지 않는 상품을 의미한다.

```
[3360, 3301, 3528, 1609, 1608, 3374, 3216, 2064, 3345, 2068]
```

그림 8. 최종 상품 후보
Figure 8. Final Product Candidates

<그림 8>은 최종적으로 필터링을 통해 도출된 상품을 나타낸다. 이 중에서 특정 상품을 선택하여 정적 추천 리스트에 포함시켜 추천을 제공한다.

이때 선택된 상품은 정적 추천 리스트가 구성하고 있는 상품들의 특성과 연관성이 있어야 하는데, 이는 추천 모델에 의해 생성된 정적 추천 리스트가 고객의 성향을 반영하고, 그 특성을 바탕으로 형성된 데이터의 집합이기 때문이다. 따라서 고객의 성향과 일치하는 상품을 선택할 필요가 있으므로 형성된 최종 상품 후보 중, 정적 추천 리스트와의 연관성이 높은 score의 상품으로 재구성 후 임의 선택한다.

이후 최종적으로 선택된 상품을 정적 추천 리스트에 추가한다. 이와 같은 방법으로 나머지 정적 추천 리스트들에 대해 동일한 작업을 수행한 후 검증을 진행한다.

$$accuracy = \frac{1}{N}\sum_{i=0}^{k} S_i \tag{7}$$

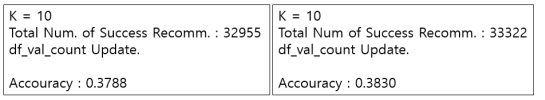
식 (7)은 정확도를 측정하기 위한 식으로 $S_i$는 $i$ 번째 검증 데이터에 대한 추천 성공 여부를 의미하며, $N$은 전체 리스트의 개수를 의미한다. 식 (7)을 통해 10개의 데이터로 형성된 86,999건의 정적 추천 리스트와 동적 추천 기법으로 형성된 리스트를 검증 데이터에 적용하여 정확도를 산출한다. 이때 Top-10 리스트 내의 상품 중 하나가 검증 데이터의 각 행에 존재하는 실제 구매 상품과 일치할 경우 추천에 성공하였다고 본다.

<표 1>은 동일한 데이터에 대해 추천 기법들을 적용하여 도출된 정확도를 나타낸다. 본 실험에서 비교 대상으로 사용된 기법은 사용자 기반 협업적 필터링(User based Collaborative Filtering)[11,12], 아이템 기반 협업적 필터링(Item based Collaborative Filtering)[13], Word2Vec[14], LSTM(Long Short Term Memory)[15] 기반 추천 모

델이며, 동일한 조건인 Top-10의 후보군을 활용하여 평가를 진행한다.

표 1. 추천 정확도 비교분석
Table 1. Recommendation Accuracy Comparison Analysis

| Reomm. Tech. / Reomm. Num. | User CF | Item CF | Word2Vec | LSTM | *This Research* |
|---|---|---|---|---|---|
| Top-10 | 0.1153 | 0.1050 | 0.1559 | 0.3788 | ***0.3830*** |




LSTM　　*This Research (Dynamic Recomm.)*

그림 9. LSTM과 동적 추천의 정확도 비교
Figure 9. Comparison of Accuracy with LSTM and Dynamic Recommendation

<그림 9>는 기존 연구 LSTM[10]과 동적 추천 기법의 정확도를 비교한다. 기존 정적 리스트의 정확도는 37.88%로 86,999개의 검증 데이터 중 32,955개의 추천 리스트가 적중하였으며, 동적 추천으로 형성된 추천 리스트의 정확도는 38.3%로 33,322개의 리스트가 적중하였다.

본 실험을 통하여 기존의 정적 추천 리스트와 유사한 추천 리스트를 제공하고, 그로부터 적절한 상품을 찾아 동적 추천을 적용한 결과 기존에 비해 정확도가 향상됨을 비교 분석하였다. 또한 재학습 없이 동적 추천을 제공하여 기존의 재학습하는 과정에 대한 추천 시스템의 아키텍처를 개선할 수 있는 기반을 제시하였다.

## 5. 결 론

본 연구에서는 기존의 추천 기법에 의한 정적 추천 성능을 향상시키고자 동적 추천 아키텍처 및 절차를 제안하였다. 본 동적 추천 아키텍처는 Elasticsearch 엔진을 기반으로 정적 추천 상품 리스트를 적용하여 동적 상품 리스트를 도출한다. 동적 상품 리스트 도출은 Elasticsearch의 BM25 알고리즘을 적용하였으며, 도출된 동적 추천 상품은 Top-10을 기준으로 정적 추천 상품과 조합하여 구성된다. 실험에서 본 연구에서 제안한 동적인 추천 아키텍처를 검증하고자 기존의 추천 기법들과 동일 거래 데이터를 적용하여 실험하였으며 기존의 추천 기법들과 비교하여 본 동적 추천 아키텍처를 적용했을 때 정확성이 향상됨을 확인하였다. 향후 연구에서는 본 동적 추천 아키텍처를 기반으로 다양한 분야의 전자상거래 추천 서비스에서 활용 가능하도록 OpenAPI 구조를 연구한다.

## References

[1] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, *Analysis of recommendation algorithms for e-commerce*, Proc. of 2nd ACM Conference on Electronic Commerce, pp. 158-167, 2000.

[2] J. Herlocker, J. A. Konstan, and J. Riedl, *An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms*, Information Retrieval, Vol. 5, No. 4, pp. 287-310, 2002.

[3] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, *Session-based recommendations with recurrent neural networks,* Proc. of ICLR 2016, pp. 1-10, 2016.

[4] G. Salton, and M. J. McGill. *Introduction to modern information retrieval,* McGraw-Hill Book Company, New York, 1983.

[5] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, *Okapi at TREC-3*, Proc. of Text REtrieval Conference, 1994.

[6] M. H. Kwon, S. E. Kong, Y. S. Choi,

*Improving recurrent neural network based recommendations by utilizing embedding matrix*, Journal of KIISE, Vol. 45, No. 7, pp. 659-666, Jul. 2018.

[7] H. Inan, K. Khosravi, and R. Socher, *Tying word vectors and word classifiers: A loss framework for language modeling,* arXiv:1611.01462, 2016.

[8] Practical BM25 - Part 2: The BM25 Algorithm and its Variables, https://www.elastic.co/kr/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables, Apr. 2018.

[9] Understanding the Inverted Index in Elasticsearch, https://codingexplained.com/coding/elasticsearch/understanding-the-inverted-index-in-elasticsearch, May 2018.

[10] C. J. Kim, J. H. Jeong, C. W. Jo, and J. K. Yoo, *A performance evaluation analysis of product recommendation techniques*, Journal of Knowledge Information Technology and Systems(JKITS), Vol. 14, No. 5, pp. 503-513, Oct. 2019.

[11] R. B. Yates, and B. R. Neto, *Modern information retrieval,* ADDISON-WESLEY, New York, 1999.

[12] J. Herlocker, J. A. Konstan, and J. Riedl, *An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms*, Information Retrieval, Vol. 5, No. 4, pp. 287-310, 2002.

[13] Q. Xu, S. Zheng, and M. Cai , *IBCF Improved algorithm based on the tag*, ICCNCE 2013, pp. 265-268, 2013.

[14] X. Rong, *Word2vec parameter learning explained,* arXiv:1411.2738 [cs], 2014.

[15] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, Journal of IEEE Transactions on Neural Networks, Vol. 5, No. 2, pp. 157-166, Mar. 1994.

## Elasticsearch 기반의 동적 추천 아키텍처 및 절차

**정지현[1], 김철진[2]**

[1] *인하공업전문대학 컴퓨터시스템과 학부생*

[2] *인하공업전문대학 컴퓨터시스템과 교수*

**요 약**

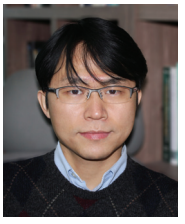
전자상거래의 상품 추천은 다양한 추천 기법을 활용한다. 일반적으로 적용하는 추천 기법은 협업적 필터링 기법인 사용자 기반 협업적 필터링과 아이템 기반 협업적 필터링 기법이 있으며, RNN(Recurrent Netural Network)과 LSTM(Long Short Term Memory)과 같은 인공지능 기술을 기반으로 하는 추천 기법을 활용한다. 이러한 추천 기법들을 통해 생성되는 추천 데이터는 데이터베이스에 저장되어 사용자 행위(검색, 주문, 등)에 의해 정적인 데이터 형태로 추천된다. 그러나 정적 추천 상품 리스트를 통한 추천은 사용자의 구매력을 향상 시키는데 한계가 있다. 추천 상품의 다양성과 실시간성을 제공하여 추천 상품 리스트의 동적 구성이 가능하다면 구매력을 더욱 향상시킬 수 있을 것이다. 이에 본 논문에서는 동적 추천을 제공하기 위한 아키텍처 및 절차를 제안한다. 제안하는 동적 추천 아키텍처는 Elasticsearch를 기반으로 정적 추천 상품을 적용하여 동적 추천 상품을 구성한다. 실험에서는 동일한 거래 데이터를 기존의 추천 기법들에 적용하여 정확성을 비교 분석하였으며, 본 논문에서 제안하는 동적 추천 기법이 기존의 추천 기법에 비해 정확성이 향상됨을 확인하였다.

## 감사의 글

이 논문은 2020년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2018R1D1A1B07049667).

**Ji Hyun Jeong** is currently studying bachelor's course in Dept. of Computer Engineering from Inha Technical College. He research focus is deep learning(product recommendation, audio recognition) and mobile service.

*E-mail address*: jhjeong@itc.ac.kr



**Chul Jin Kim** received both M.S. and Ph.D. degree in Computer Science from the Soongsil University in 1998 and 2004. He worked as a senior researcher in SAMSUNG Electronics from 2004 to 2009. He has been a professor in the Department of Computer Systems and Engineering at Inha Technical College since 2009. He research focus is software engineering, object-oriented & component technique, software architecture, customization, and deep learning.

*E-mail address*: cjkim@inhatc.ac.kr