레코드 연결을 위한 속성인지 메타블로킹

이주현○, 김현호, 강인호
네이버
dev_joohyun.lee@navercorp.com, kim.hh@navercorp.com, once.ihkang@navercorp.com

Property-aware Meta Blocking for Record Linkage

Joo-Hyun Lee○, Hyun-Ho Kim, In Ho Kang
Naver Corp.

요 약

레코드 연결의 대표적인 문제 중 하나는 레코드 간 비교 비용이 크다는 것이다. 이러한 문제를 해결하기 위해서는 레코드 연결에 필수적으로 블로킹 단계가 포함되어야 한다. 블로킹이란 같은 레코드일 가능성이 높은 대상들을 그룹화하여 비교연산을 수행할 대상을 선정하는 단계를 말한다. 블로킹의 목적은 최대한 결과의 recall을 희생시키지 않으면서 비교 연산 횟수 최소화하는 것이다. 메타 블로킹은 가중치 그래프를 블로킹에 적용함으로써 전통적인 블로킹 방식의 한계를 극복하고 더 좋은 성능을 나타내는 모델이다. 본 논문에서는 메타블로킹에서 주목하지 않았던 블록 생성방식을 데이터베이스 속성에 따라 블록을 생성하는 방식으로 개선하고 그에 맞는 가중치 계산식을 제안하였다. 또한 키 기반 블로킹, 메타블로킹, 속성인지 메타블로킹으로 생성된 블로킹 결과에 대한 성능을 측정 및 비교하였다.

주제어: 레코드 연결, 데이터베이스 연결, 블로킹

1. 서론

최근 IT기술의 발전으로 데이터 수집이 보편화되면서 수집된 대용량 데이터를 다루는 빅데이터의 필요성이 강조되고 있다. 이에 따라, 다른 데이터 원천에서 생성된 다수의 데이터베이스에서 같은 대상을 연결하는 Record Linkage(레코드 연결) 기술의 중요성과 필요성이 증가하고 있다[1].

Record(레코드)란 DB에서 하나의 data 또는 row를 의미하며, 레코드 연결이란 2개 또는 2개 이상의 데이터베이스에서 동일한 레코드를 찾아내거나, 하나의 데이터베이스에서 동일한 레코드를 찾아내는 기술을 말한다[2]. 레코드 연결은 접근 방식에 따라 DB Linkage, Entity Resolution, Entity Matching 등의 명칭으로 불린다. 본 논문에서는 Record Linkage(레코드 연결)를 사용한다.

레코드 연결의 대표적인 문제 중 하나는 efficiency(효율성)문제이다. n개의 레코드를 가진 d개의 DB를 연결하는 레코드 연결에서 같은 레코드를 찾기 위해서는 각 레코드를 다른 DB의 모든 레코드들과 비교해야 한다. 그렇기에 레코드 연결은 $O(n^d)$의 complexity(복잡도)를 가진다. 이는 DB의 수(d)와 DB에 들어 있는 레코드의 수(n)에 따라 성능상의 병목을 유발한다. 이러한 성능상의 병목을 해소하기 위해 레코드 연결은 품질에 영향을 주지 않으며 레코드간 비교 횟수를 감소시켜야 한다. 이를 해결하기 위해 레코드 연결은 각 레코드를 Block(블록)으로 그룹화하는 기법인 Blocking(블로킹) 기법을 수행한다[3].

가장 전통적인 블로킹 방식은 Key-Based(키 기반) 블로킹이다. 키 기반 블로킹이란 레코드를 대표하는 속성의 값이 동일한 대상들을 그룹화하여 블록을 생성하는 방식이다. 예를 들어 인물DB에서는 '이름' 속성이 이용될 수 있고, 앨범 DB에서는 '앨범 명' 속성이 이용될 수 있다. 키 기반 블로킹 방식은 직관적이며 빠르게 결과를 도출할 수 있지만 몇 가지 한계점을 가지고 있다. 첫 번째 한계점은 표현방식의 차이로 인한 누락 문제이다. 동일한 대상을 가리키지만 표현의 방식이 다른 경우 블록에서 누락될 수 있다. 예를 들어 '아이유' 와 'IU' 는 동일한 대상을 지칭하지만, 동일 블록으로 그룹화되지 않는다. 두 번째 한계점은 레코드를 나타내기 위해 다수의 속성이 필요한 경우이다. 하나의 속성만으로 블로킹이 불가한 경우, 키 기반 방식은 다수의 속성을 블록의 키로 사용해야 한다. 이 때 속성을 키로 사용하는 방식에 따라서 많은 블록의 생성으로 효율성이 감소하거나, 적은 블록의 생성으로 다수의 누락이 발생할 수 있다. 이러한 키 기반 블로킹의 한계를 극복하기 위해 다양한 방식이 연구되고 있다.

Meta-Blocking(메타블로킹)은 키 기반 블로킹 방식과 다르게 특정 속성이 아닌 모든 속성의 값을 사용하여 블록을 생성하며, 생성된 정보에서 블록을 공유하는 레코드 간의 관계를 가중치 그래프 형태로 변환한다. 블록은 DB의 전체 속성을 사용하고 속성 중 하나라도 값을 공유하면 두 개의 레코드는 간선을 가지게 된다. 그 다음 그래프의 간선에 가중치를 부여하고 가중치에 따라 불필요한

간선을 Pruning(가지치기)한다. 메타블로킹은 '두 레코드가 많은 블록을 공유할수록 같은 레코드일 확률이 높다' 는 것을 전제로 전체 블록에서 불필요한 블록을 제거하는 방식으로 최종 블로킹 결과를 생성한다. 이와 같은 방식으로 메타블로킹은 키 기반 블로킹 방식의 한계를 극복할 수 있다. 한 속성 값의 표현방식이 다르더라도 다른 속성의 값을 통해 같은 블록에 포함된다. 또한, 레코드를 표현하기 위해 다수의 속성이 필요한 경우도 효과적으로 블로킹 할 수 있다. 단, 메타블로킹은 DB의 속성을 고려하지 않고 모든 속성의 값을 키로 사용하여 블록을 생성한다. 그러나 실제 DB의 속성은 유사한 정보들이 집약되어 있다는 특성을 가진다.

본 논문에서는 DB의 속성이 가지는 특성을 활용한 새로운 블록 생성방식과 블록 생성방식이 개선됨에 따라 변경되는 가중치 그래프에서 사용할 수 있는 새로운 가중치 계산식을 제안함으로써 메타블로킹 모델을 개선한 속성인지 메타블로킹 모델을 제안한다. 평가 데이터는 음원서비스의 앨범 정보를 사용하며 키 기반 블로킹 모델, 메타블로킹 모델, 속성인지 메타블로킹 모델을 3가지 지표를 사용하여 평가한다. 평가지표는 recall score을 나타내는 Pairs Completeness 와 precision을 나타내는 Pairs Quality, 베이스라인 모델 대비 효율성 증가를 표현하는 Reduction Ratio를 사용한다[4][5].

2. 관련 연구

레코드 연결에서 블로킹은 연결 결과의 품질과 수행 시간에 큰 영향을 미치기 때문에 블로킹에 대한 많은 연구가 진행되며 다양한 모델이 제시되고 있다.

메타블로킹은 가중치 그래프를 사용하여 블로킹 방식을 개선하였다[6]. 메타블로킹은 먼저 블록을 생성하고, 생성된 블록 정보를 가중치 그래프로 변환한다. 각 간선에 가중치를 부여하기 위한 5가지의 가중치 계산식을 제시하고 있으며, 이후 가중치를 활용하여 불필요한 간선을 제거한다. 위 과정을 통해 최종 블로킹 결과를 생성한다. 가중치 그래프의 노드는 레코드를, 두 개의 노드를 연결하는 간선은 동일한 블록에 속하고 있음을, 가중치는 두 레코드가 동일한 레코드일 확률을 나타낸다. 메타블로킹은 가중치 계산식과 불필요한 간선을 선별하는 방식을 중점적으로 다룬다는 점에서 블록 생성 방식에 초점을 두어 개선방안을 제시하는 본 논문과는 차이가 존재한다. 또한 메타블로킹 모델에 지도학습의 개념을 적용한 연구도 발표되었다[7]. 해당 논문에서는 그래프의 가중치를 공유하는 블록의 빈도로 표현하지 않고 특정 feature vector(특징 벡터)로 표현한다. 이후 훈련 데이터를 생성하고 이진 분류 모델을 사용하여 특징 벡터로 표현된 각 간선의 유지 또는 삭제 여부를 판단한다. 이는 가중치를 활용한 가지치기와 동일한 결과를 가져오게 되고 기존 메타블로킹과 동일한 형태의 최종 블로킹 결과를 생성한다.

이 밖에도 최근 주목받고 있는 인공신경망을 적용한 AutoBlocking(오토블로킹) 방식이 제안되었다[8]. 오토블로킹은 크게 5단계로 구성된다. DB의 각 속성에 token embedding(토큰 임베딩)과 attention-based attribute embedding(어텐션 기반 속성 임베딩)을 적용하여 각 속성을 벡터로 변환하고 tuple signature(튜플 임베딩)을 통해 레코드 단위의 벡터로 통합한다. 이후 훈련 데이터를 활용하여 어텐션 기반 속성 임베딩 모델과 튜플 임베딩의 가중치를 수정한다. 훈련이 끝난 뒤에는 근접이웃 알고리즘을 통해 최종 블로킹 결과를 생성한다.

3. 개선 방안

본 논문은 기존 메타블로킹의 블록 생성방식을 개선한 속성인지 메타블로킹 모델을 제안한다. 블록 생성방식의 개선은 가중치 그래프에 변경을 가져오므로 변경된 그래프에 적합한 새로운 간선 가중치 계산식도 제안한다.

3.1 블록 생성방식 개선

기존 메타블로킹에서는 블록을 생성할 때, DB의 속성을 구분하지 않고 모든 속성에 들어있는 값을 이용하여 같은 값을 가지고 있는 레코드들을 하나의 블록에 포함시키는 방식으로 블록을 구성한다. 이러한 블록 생성방식의 한계는 동일한 '값' 과 동일한 '의미' 를 구분할 수 없다는 것이다. 속성에 따라 동일한 값이 동일한 의미를 가지지 않을 수 있다. 예를 들어 '나이' 속성 값 30과 '출연 횟수' 속성 값 30은 동일한 값을 가지지만 동일한 의미를 가지지 않는다.

본 논문에서는 동일하거나 유사하다고 판단되는 속성들을 선별하고 그룹화한 뒤 동일 그룹 속성에서만 블록을 생성하여 값이 아닌 의미가 동일한 경우에 블록을 생성하는 방식을 제안한다. 그룹화된 속성으로 구분된 블록을 생성하기 위해서 각 DB에서 동일하거나 유사한 속성을 선별해야 한다. 본 논문에서는 연구자가 직접 속성을 그룹화하였다. 동일하거나 유사한 속성이라 함은 이종의 DB에서 유사한 주제에 대한 정보를 가지는 속성을 의미한다. 앨범 DB에서는 [DB1의 앨범제목 속성, DB2의 앨범제목 속성], [DB1의 앨범 발매일 속성, DB2의 앨범 발매일 속성]을 동일하거나 유사한 속성의 예시로 들 수 있다.

블록을 구성할 속성 그룹을 선정한 뒤, 각 DB에서 그룹화된 속성에 해당하는 값을 사용하여 블록을 생성한다. 블록에 레코드가 포함되는 방식은 기존 메타블로킹과 동일하지만, 블록을 생성하는 모수가 전체 속성이 아닌 그룹화된 속성인 점이 기존 모델과 다르다. 그림 1은 본 논문에서 사용한 블록 생성방식의 구체적인 알고리즘을 보인다.

Algorithm 1 Building Block Collection

Input: selected DB property groups Gs, DB_a, DB_b
Output: Block Collections B

```
n = 0                                              ▷ dict key to separate property groups.
Dict_B = key-value dict that represent Block Collection.
for all property group p_a, p_b ∈ G do
    for all Record ∈ DB_a do
        v = Record[p_a]                            ▷ value corresponding to p_a
        if v not in Dict_B[n] then
            Dict_B[n][v] = Set()
        end if
        Dict_B[n][v].add(Id of Record)
    end for

    for all Record ∈ DB_b do
        v = Record[p_b]                            ▷ value corresponding to p_b
        if v not in Dict_B[n] then
            Dict_B[n][v] = Set()
        end if
        Dict_B[n][v].add(Id of Record)
    end for
    n = n + 1
end forreturn Tree
```

그림 1. 속성 그룹화 블록 생성 알고리즘

3.2 Common Blocks with Inverse Importance Scheme 제안

기존 메타블로킹에서는 5가지의 간선 가지치기 식을 제안했다. 또한 5가지 식에서 Enhanced C ommon Blocks Scheme(ECBS)과 Enhanced Jaccard Scheme(EJS)이 가장 좋은 성능을 보였음을 밝혔다[6].

ECBS는 두 레코드가 공통으로 속한 블록의 수에 전체 블록 수를 각 레코드가 속한 블록의 수를 나눈 값을 곱하여 가중치를 계산한다. 공통으로 가지고 있는 블록의 수를 사용하되, 레코드가 여러 개의 블록에 포함될 수록 불필요한 블록이 많다는 IDF(Inverse Document Frequency) 개념을 도입한 가중치 계산식이다.

EJS는 두 레코드의 블록 정보를 통해 자카드 유사도를 구한 뒤, 전체 간선의 수를 각 레코드에 연결된 간선의 수로 나눈 값을 곱하여 가중치를 계산한다. 이는 블록이 아닌 노드와 간선의 정보를 활용한다는 점에서 ECBS와 다르지만, 근본적으로 레코드가 많은 레코드와 연결될수록 불필요한 연결이 많다는 IDF 개념으로 가중치를 조정한다는 점에서 ECBS와 유사하다.

본 논문은 속성 그룹화를 통해 블록 생성과정에서 불필요한 블록을 최소화하는 블록 생성방식을 제안했다. 이로 인해 하나의 블록에 포함되는 레코드의 수와 불필요한 블록의 수가 감소하게 된다. 이렇게 변경된 블록 정보는 가중치 그래프 전체 간선의 수가 줄어드는 효과를 가져오게 된다.

본 논문에서는 변경된 그래프에 적합한 가중치 산출을 위하여 공통 블록만을 사용하여 가중치를 계산하는 새로운 계산식 Common Blocks with Inverse Importance Scheme(CBIIS)을 제안한다. CBIIS는 공통 블록의 수를 사용하되, 공통 블록의 중요도를 통해 가중치를 조정한다. 수식으로 나타낸 CBIIS 식은 수식 1과 같다.

수식 1에서 $|B_{i,j}|$은 레코드 i와 j가 공통으로 포함되어 있는 블록의 개수를 의미하며 $\|b\|$은 블록에 속한 레코드의 개수를 의미한다. $|N|$은 전체 노드의 수를 의미한다.

$$e_{i,j} \cdot weight = |B_{i,j}| \cdot \log\left(\frac{\sum_{b_k \epsilon B_{i,j}} \frac{|N|}{\|b_k\|}}{|B_{i,j}|}\right)$$

수식 1. 가중치 계산식 CBIIS

CBISS은 공통 블록의 수를 사용하지만 가중치 조정을 레코드 정보가 아닌 공통 블록의 정보만을 사용한다는 점이 ECBS과의 차이점이다. 가중치 조정은 공통 블록에 포함되어 있는 각 블록 중요도의 평균값에 log연산을 수행한 값을 이용한다. 이 때, 각 블록의 중요도는 전체 노드의 수에 블록에 속한 레코드의 수를 나눈 값으로 산출한다. IDF 개념에서 착안하여 블록에 많은 레코드가 포함될수록 블록의 중요도가 낮다는 접근을 통해 분모를 블록에 속한 레코드의 수로 선정하였다. 예를 들어 다수의 가수가 참여한 앨범은 'Various Artists'로 가수명이 표기된다. 이러한 가수명으로 생성된 블록의 중요도는 비교적 낮다고 판단할 수 있고, 이는 블록에 속한 레코드의 수를 분모로 사용함으로써 수치에 반영된다. 실제로 평가 데이터에서 'Various Artists' 블록에는 627개의 레코드가 포함되었고 '아이유' 블록에는 246개의 레코드가 포함되었다. 이 때 각 블록의 중요도는 150.26과 381.15로 산출되었다. 이를 통해 수식1에서 각 블록의 중요도가 적절히 계산되고 있음을 확인할 수 있다. 분자는 동일한 값을 제공함과 동시에 충분히 큰 값을 제공하여 중요도의 유의미한 차이를 도출하기 위하여 전체 노드의 수를 사용하였다.

4. 성능 평가

4.1. 실험 환경

본 논문에서는 음원서비스 Vibe에서 제공받은 앨범 DB와 음원서비스 Bugs의 공식페이지에 공개된 정보를 통해 생성한 앨범 DB를 이용하여 성능을 평가하였다[9][10]. 좋아요 개수를 이용한 필터링을 통해 Vibe DB는 18,113개의 레코드를 가지며 Bugs DB는 75,651개의 레코드를 가진다. 블로킹의 목적은 최종 정답으로 예측될 가능성이 있는 후보군을 선별하는 것이기 때문에 일반적인 평가 지표인 recall, precision 등의 계산식을 사용하는 것은 적절하지 않다. 이에 블로킹 결과 평가에 주로 사용되는 3가지 지표를 통해 성능을 평가한다[4][5]. D는 전체 정답쌍을 의미하고 $|D|$는 D에 포함된 레코드 쌍의 개수를 의미한다. $D(B)$는 블로킹 결과에서 하나 이상의 블록을 공유하면서 실제로도 동일한 레코드 쌍의 집합을 의미하며, $|D(B)|$는 $D(B)$에 포함된 레코드 쌍의 개수를 의미한다. 사용한 평가지표는 아래와 같다.

(i) Pairs Completeness (PC)는 recall score를 나타내는 지표로서 전체 정답 레코드 쌍에서 블로킹 결과에서 하나 이상의 블록에 포함되어 찾아질 수 있는(detectable) 정답 레코드 쌍의 비율을 [0,1] 범위로 표현한다. 높은 PC 값은 블로킹 결과가 많은 정답 쌍을 포함하고 있음을 나타낸다[5]. 수식으로 나타낸 PC는 수식 2와 같다.

$$\mathrm{PC}(B) = \frac{|D(B)|}{|D|}$$

수식 2.

(ii) Pairs Quality (PQ)는 precision score를 나타내는 지표로서 블로킹 생성결과에 포함되어 있는 전체 쌍에서 블로킹에 성공한 실제 정답 쌍의 비율을 [0,1] 범위로 표현한다. 높은 PQ 값은 전체 블로킹 결과에서 정답 쌍이 차지하는 비율이 높음을 의미한다[5]. 수식으로 나타낸 PQ는 수식 3와 같다.

$$\mathrm{PQ(B)} = \frac{|D(B)|}{\|B\|}$$

수식 3.

(iii) Reduction Ratio (RR)는 베이스라인 모델이 생성한 전체 비교 대상 중 평가 모델을 통해 제거된 비교 쌍의 비율을 [0,1] 범위로 표현한다. RR 값이 높다는 것은 불필요한 비교가 많이 제거되었다는 것을 의미한다[5]. 수식으로 나타낸 RR은 수식 4와 같다.

$$\mathrm{RR(B}, B_{baseline}) = 1 - \frac{\|B\|}{\|B_{baseline}\|}$$

수식 4.

세 가지 평가 지표 모두 높을수록 모델의 성능이 우수하다는 것을 의미한다. 그러나 $|D(B)|$의 증가는 $\|B\|$의 급격한 증가를 가져오기 때문에 PC 지표와 PQ, RR 지표는 트레이드 오프 관계를 가진다. 그렇기에 지표 간의 균형을 유지하는 것도 중요한 평가 요소이다[4].
평가대상 모델은 '앨범 명' 속성을 키로 수행한 키 기반 블로킹 모델, 메타블로킹 모델, 속성인지 메타블로킹 모델로 총 3가지이다. 단, 데이터의 특성상 모든 레코드가 정답을 가질 수 없으므로 PQ 값은 전체 블록에 대한 값(전체대상 기준 PQ)과 정답이 포함된 블록에 대한 값(일치대상 기준 PQ) 두 가지를 산출하였다. 또한 RR 지표는 비교를 위한 모델이 필요하므로 모든 레코드를 비교하는 brute force(완전 탐색)모델과 키 기반 모델을 비교 모델로 사용하였다.

4.2 실험 결과 - 모델 비교

표 1은 본 논문에서 평가 대상으로 한 세 가지 모델을 비교평가한 결과를 보인다. 표 1의 결과에서 PC 지표는 3개 모델 모두 90% 이상으로 비교적 높은 수치 값을 보였고, 그 중에서도 속성인지 메타블로킹 모델이 가장 높은 수치 값을 보였다. 그러나 키 기반 모델은 PQ 값이 다른 모델에 비해 현저히 낮다. 이는 키 기반 블로킹 결과가 많은 정답 쌍을 포함하지만 동시에 많은 불필요한 비교 쌍을 포함하고 있어 높은 비교 비용이 발생할 수 있음을 의미한다[5]. 반면 속성인지 메타블로킹 모델은 가장 높은 PC 값을 보임과 동시에 가장 높은 PQ 값을 보였다. 이는 실제 정답 쌍을 가장 많이 포함하면서도 불필요한 쌍은 가장 적게 포함하고 있음을 의미한다[5]. RR 지표 값은 3가지 모델 모두 완전 탐색 모델과의 비교 수치 값은 99.99%로 뚜렷한 성능 개선을 보인다. 실질적인 모델의 개선 결과로 볼 수 있는 키 기반 모델과의 비교 수치 값은 속성인지 메타블로킹 모델이 메타블로킹 모델보다 약 3% 높은 성능을 보였다. 속성인지 메타블로킹 모델이 세 가지 지표에서 모두 높은 값을 보여 뚜렷한 성능 향상을 보였음을 알 수 있다[4].

<표 1> 모델 평가지표 비교

	PC (%)	전체 대상 기준 PQ (%)	일치 대상 기준 PQ (%)	키 기반 기준 RR (%)	완전 탐색 기준 RR (%)
키 기반 모델	90.83	9.46	68.93	-	99.99
메타 블로킹	92.16	27.8	**99.83**	65.48	99.99
속성 인지 메타 블로킹	**94.31**	**30.88**	99.55	**68.19**	99.99

4.3 실험 결과 - 가중치 계산식 비교

표 2는 본 논문에서 제시한 가중치 계산식인 CBIIS와 기존 메타블로킹 모델 가중치 계산식인 ECBS와 EJS을 비교 평가한 결과를 보인다.

<표 2> 가중치 계산식 평가지표 비교

	PC (%)	PQ (%)	RR (%)
CBIIS	**94.31**	**30.88**	68.19
ECBS	92.8	30.51	**68.32**
EJS	93.17	25.9	62.55

표 2의 결과에서 CBIIS 계산식이 가장 높은 PC, PQ 값을 보였고 ECBS 계산식이 가장 높은 RR 값을 보였다. 그러나 CBIIS이 ECBS보다 PC 값은 약 1.5%가 높으면서 PQ, RR 스코어의 차이는 0.3% 이하로 작은 차이를 보인다. 이는 적은 비교 비용의 증가로 많은 정답이 포함되었음을 의미한다[5]. EJS와 비교해서는 3가지 지표 값 모두 CBIIS가 더 높은 값을 보였다. 이러한 실험 결과를 통해 CBIIS가 가장 좋은 성능을 보였음을 알 수 있다.

5. 결론 및 향후 연구

본 논문은 DB 속성 그룹화를 진행한 뒤, 블록을 생성하는 방식으로 기존의 메타블로킹의 블록 생성과정을 개선하였다. 또한 블록 생성방식 개선으로 인해 변경된 가중치 그래프에 적합한 새로운 가중치 계산식 CBIIS를 제안하였다. 뿐만 아니라 블로킹 결과를 평가하는데 주로 사용되는 3가지 지표를 활용하여 모델을 평가하고 성능이 개선되었음을 보였다. 속성인지 메타블로킹 모델을 통해 블로킹 단계의 성능을 향상시킴으로써 최종적으로는 레코드 연결 결과의 품질 향상과 커버리지를 확대할 수 있다.

향후 연구로는 사용자의 개입을 최소화하기 위해 속성 그룹화 단계를 자동화하는 방안에 대한 연구가 필요하다. 또한 현재 블록 생성을 위한 값 매칭 방식은 완전 일치 방식을 사용하고 있지만, 오타 또는 일부 특수 문자 등에 의한 누락을 방지할 수 있는 새로운 값 매칭 방식 적용에 대한 연구가 필요하다.

참고문헌

[1] Sayers, A drian, et al. "Probab ilistic record linkage." International j ournal of epidemiology 45.3 (2016): 954-964.

[2] Winkler, William E. "The state of record linkage and current research problems." Statistical Research Division, US Census Bureau. 1999.

[3] Steorts, Rebecca C ., et al. "A comparison of blocking methods for record linkage." International conference on privacy in statistical databases. Springer, Cham, 2014.

[4] Papadakis, George, et al. "Comparative analysis of approximate b locking techniques for entity resolution." Proceedings of the VLDB Endowment 9.9 (2016): 684-695.

[5] Christen, Peter. "A survey of indexing techniques for scalable record linkage and deduplication." IEEE transactions on knowledge and data engineering 24.9 (2011): 1537-1555.

[6] Papadakis, George, et al. "Meta-blocking: Taking entity resolution to the next level." IEEE Transactions on Knowledge and Data Engineering 26.8 (2013): 1946-1960.

[7] Papadakis, G eorge, G eorge P apastefanatos, and Georgia Koutrika. "Superv ised meta-blocking." Proceedings of the V LDB Endowment 7.14 (2014): 1929-1940.

[8] Zhang, Wei, et al. "AutoBlock: A hands-off blocking framework for entity matching." Proceedings of the 13th International Conference on Web Search and Data Mining. 2020.

[9] 음원사이트 Vibe, https://vibe.naver.com

[10] 음원사이트 Bugs, https://www.bugs.co.kr