확장 학습 블룸 필터의 효율적인 구현

(Efficient Implementation of Extended Learned Bloom Filter)

요 약 블룸 필터는 집합을 표현하는 자료구조로 데이터의 포함 여부에 대해서 반환하는 역할을 수행한다. 단, 공간을 적게 사용하는 대가로 거짓 양성을 반환하는 경우가 존재한다. 학습 블룸 필터는 기존의 블룸 필터에 추가적으로 모델을 전처리 과정에 사용하여 거짓 양성 비율을 개선하는 방법이다. 본 논문에서는 모델을 전처리 과정 뿐만 아니라 학습 해시 함수로 사용하는 학습 블룸 필터를 구현하며 이를 확장 학습 블룸 필터로 부른다. 확장 학습 블룸 필터는 보조 필터의 공간 일부를 변수 α를 통해서 학습 해시 함수로 사용하여 거짓 양성 비율을 개선한다. 추가적으로, 확장 학습 블룸 필터를 구현하며 발생했던 거짓 음성 오류의 원인과 해결방안을 소개하며, 모델로 사용되는 인공신경망의 구조를 조정하여 거짓 양성 비율을 개선할 수 있음을 실험적으로 보인다.

키워드 : 자료구조, 블룸 필터, 학습 블룸 필터, 학습 해시 함수, 학습 인덱스

Abstract Bloom filter is a data structure that represents a set and returns whether data is included or not. However, there are cases in which false positives are returned at the cost of using less space. The learned bloom filter method applies a model as a pre-processing process of the existing bloom filter to improve the false positive rate. In this paper, we implement a learned bloom filter that uses the model not only as a pre-processing process but also as a learned hash function, which is called an extended learned bloom filter. The extended learned bloom filter improves the false positive rate by using the spatial part of the auxiliary filter as a learned hash function via the parameter α. Additionally, we introduce the causes and solutions of false negative errors that occurred while implementing the extended learned bloom filter, and experimentally show that the false positive rate can be improved by adjusting the structure of the artificial neural network used as the model.

Keywords : Data Structure, Bloom Filter, Learned Bloom Filter, Learned Hash Function, Learned Index

1. 서론

블룸 필터[1]는 1970 년 B. H. Bloom 에 의해서 제안된 확률적인 자료구조이다. 블룸 필터는 집합을 표현하며, 질의된 원소가 집합 내부에 존재하는지 확인하는 기능을 수행한다. 블룸 필터는 집합 S 를 표현하기 위해서 k 개의 해시 함수, m 크기의 비트 벡터를 사용한다. k 개의 해시 함수는 원소가 저장될 비트 벡터의 인덱스 값을 반환한다. 비트 벡터의 공간은 1 또는 0 으로 표현되고, 데이터의 존재 여부를 표현할 때 사용된다. 즉, 블룸 필터는 해시 함수와 비트 벡터를 함께 사용하여 집합 S 의 원소들에 대한 존재 여부를 확인하는 기능을 수행한다.

블룸 필터를 생성하기 위해서는 집합 S 에 있는 각 원소에 대해서 k 개의 해시 함수를 통해서 얻은 비트 벡터의 인덱스 값을 1 로 갱신하면 된다. 질의하는 원소가 k 개의 해시 함수에 대해서 모두 1 이라는 비트 벡터의 값을 가지게 되면 해당 원소는 집합에 포함됨을 의미하고, 한번이라도 0 이라는 비트 벡터의 값을 가지게 되면 집합에 포함되지 않음을 의미한다. 허나, 집합에 포함되어 있지 않는 원소임에도 불구하고 가끔씩은 집합에 포함되어 있음을 반환하는 경우가 존재하며, 이를 거짓 양성이라고 한다. 거짓 양성은 제한된 m 크기의 비트 벡터에 표현하고자 하는 집합 S 의 크기가 커질수록 빈번하게 발생한다. 거짓 양성이 발생함에도 불구하고, 블룸 필터는 집합 S 를 표현하는데 적은 양의 공간을 사용하기 때문에 여러 분야에서 활발하게 사용되고 있다.

예를 들어, Monkey[2]는 LSM 트리 기반의 키-값 데이터베이스이다. Monkey 는 키-값의 존재 여부를 효율적으로 확인하기 위해서 블룸 필터를 사용한다. LSM 트리에 블룸 필터를 사용하지 않았다면, 각 키-값을 조회하기 위해서 디스크 I/O 가 발생했을 것이다. 하지만, 블룸 필터가 양성을 반환했을 때만 디스크 I/O 를 수행하여 전체적인 데이터베이스의 성능을 개선시킨다.

CRLite[3]의 경우에는 HTTPS 지원을 위한 기관 인증서 정보를 계층적인 블룸 필터를 통해서 사용할 것을 제안한다. CRLite 의 사용으로 여러 개의 인증서 정보를 하나의 계층적인 블룸 필터로 표현하여, 공간 효율적으로 인증서 정보를 확인할 수 있게 된다.

학습 블룸 필터[4]는 2018 년 T. Kraska 에 의해서 제시된 블룸 필터의 변형이다. 학습 블룸 필터는 집합의 포함 여부 문제를 데이터 과학 문제로 해석하여, 이진 분류 모델을 전처리 과정에 사용할 것을 제안한다. 학습 블룸 필터는 모델, 임계치 그리고 보조 필터로 구성된다. 전처리 과정에 사용되는 모델은 [0, 1]사이의 값을 반환한다. 해당 값이 임계치보다 크면 집합에 포함됨을 의미하고 임계치보다 같거나 작으면 집합에 포함되지 않음을 의미한다. 이후에 모델이 분류하지 못한 원소들은

보조 필터에 의해서 집합의 포함 여부를 반환하게 된다. 전처리 과정인 모델에서는 거짓 음성을 반환될 수 있지만, 블룸 필터와 동일한 역할을 수행하는 보조 필터에 의해서 거짓 음성이 발생하지 않게 된다.

학습 해시 함수[4]는 2018 년 T. Kraska 에 의해서 제안된 해시 함수이다. 이는 모델의 출력값인 [0, 1]사이의 값을 비트 벡터의 크기인 m 으로 곱하여, 모델의 출력값을 비트 벡터의 인덱스로 매핑하는 방법이다. 즉, 모델을 f 그리고 보조 블룸 필터의 크기를 m 으로 두면 ⌊f(x) * m⌋을 통해서 비트 벡터의 인덱스를 계산할 수 있다. 여기서 x 는 블룸 필터의 입력 또는 질의되는 원소를 의미한다. 일반적인 해시 함수는 원소를 균일하게 분배해서 성능을 개선한다면, 학습 해시 함수는 동일하게 분류된 원소간의 충돌을 늘려서 성능을 개선하였다.

2. 관련 연구

Adaptive Learned Bloom Filter[5]에서는 Ada-BF 와 Disjoint Ada-BF 라는 새로운 자료구조를 소개한다. Ada-BF 의 경우에 데이터의 분포에 따라서 해시 함수의 개수를 변경하여 거짓 양성 비율을 개선한다. 학습 블룸 필터의 전처리 과정에 위치한 모델의 출력값은 [0, 1]사이로 해당 값이 높은 경우는 원소가 존재할 확률이 높다는 의미이며, 해당 값이 낮은 경우에는 원소가 존재할 확률이 낮다는 의미이다. 따라서 원소가 집합에 존재할 확률이 높은 경우에는 하나의 해시 함수를 사용하고, 원소가 집합에 존재할 확률이 낮은 경우에는 여러 개의 해시 함수를 사용하여 원소의 포함 여부를 표현한다.

Disjoint Ada-BF 의 경우에는 데이터 분포를 사용한다는 발상은 동일하지만, 기존에 하나로 사용되던 보조 필터를 여러 개의 보조 필터로 나눠서 사용하는 방법을 제안한다. 즉, 모델에서 반환되는 숫자의 범위인 [0, 1]을 g 개의 그룹으로 균일하게 나누고 그룹에 속하는 원소가 많은 경우에는 크기가 큰 보조 필터를 사용하고, 그룹에 속하는 원소가 적은 경우에는 크기가 작은 보조 필터를 사용한다. 위의 방법은 m 으로 제한된 비트 벡터의 공간을 효율적으로 사용하여, 거짓 양성 비율을 개선하는 방법이다. 단, 위에서 설명한 g 개의 그룹과 k 개의 해시함수의 숫자는 휴리스틱한 방법을 사용하여 설정한다. 따라서, 각 하이퍼파라미터 g 와 k 에 대해서는 사용자가 입력한 범위 내에서 직접 Ada-BF 또는 Disjoint Ada-BF 를 반복적으로 생성해가며 최적의 거짓 양성 비율을 갖는 인스턴스를 찾는다. 위와 같이 여러 개의 Ada-BF 를 생성하고, 최적의 Ada-BF 를 찾을 수 있는 이유는 입력될 데이터 분포를 사전에 알고 있기 때문이다. 학습 블룸 필터는 정적인 집합에 대해서만 실용적으로 사용이 가능하여, 학습 블룸 필터의 변형인 Ada-BF 와 Disjoint Ada-BF 도 위의 가정을 따른다.

Partitioned Learned Bloom Filter[6]는 Disjoint Ada-BF 의 후속 연구이다. 이는 하이퍼파라미터 g 를 설정하는데 사용한 휴리스틱한 방법을 수학적인 방법으로 대체하여 거짓 양성 비율을 개선한다. 위 논문은 g 개의 그룹을 나누기 위한 구간과 각 그룹에서 사용되어야 하는 비트 벡터의 크기를 최적화 이론을 사용하여 수식으로 표현한다. 제안된 방법인 Partitioned Learned Bloom Filter 를 사용하면 Disjoint Ada-BF 에 비해서 개선된 거짓 양성 비율을 가진다.

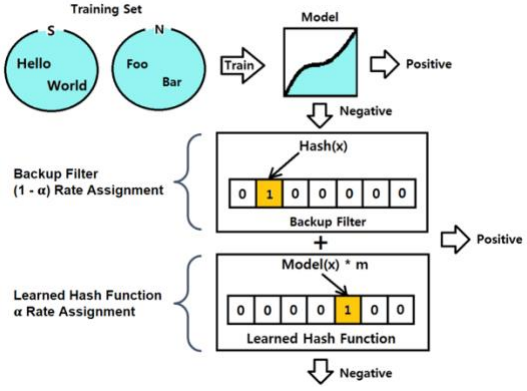

그림 1 확장 학습 블룸 필터

Fig. 1 Extended Learned Bloom Filter

3. 확장 학습 블룸 필터

본 논문에서 제안하는 모델인 확장 학습 블룸 필터는 학습 해시 함수와 기존의 블룸 필터를 함께 사용하기 위해서 α라는 변수를 추가한다. α라는 변수는 [0, 1] 사이의 숫자를 가지며, 전체 비트 벡터 중에 학습 해시 함수가 사용되는 비율을 나타낸다. 즉, 확장 학습 블룸 필터는 학습 해시 함수를 사용하는 비트 벡터의 크기를 ⌈α * m⌉ 사용하고, 기존의 보조 필터로 사용하는 비트 벡터의 크기를 ⌊(1 − α) * m⌋ 사용한다. 이는 ⌈α * m⌉ 크기의 비트 벡터(B_L)는 학습 해시 함수를 사용하고 ⌊(1 − α) * m⌋크기의 비트 벡터(B_B)는 일반적인 해시 함수가 사용되는 것을 의미한다.

블룸 필터의 집합의 크기 n 와 거짓 양성 확률 p 가 주어지면, 최적의 해시 함수 개수 k 와 비트 벡터의 크기

m 을 구할 수 있다. 위에서 구한 비트 벡터의 크기 m 에 대해서 α 비율만큼의 공간을 학습 해시 함수로 분배한다. 보조 필터의 공간은 α 비율 만큼 줄어들었기 때문에 반환되는 거짓 양성 오류는 분배 이전보다 많아지게 된다. 따라서, 학습 해시 함수가 거짓 양성 비율을 낮추는데 일조하지 않는다면 기존의 학습 블룸 필터보다 개선된 거짓 양성 비율을 갖지 못한다.

3.1 거짓 양성 보장

확장 학습 블룸 필터가 블룸 필터와 동일한 기능을 수행하기 위해서는 거짓 음성이 반환되지 않아야 한다. 이를 증명하기 위해서 확장 학습 블룸 필터에 거짓 음성이 발생했다고 가정해본다. 거짓 음성이 발생했다는 것은 양성 데이터에 대해서 음성의 결과가 반환된 것이다. 확장 학습 블룸 필터의 전처리 과정으로 사용되는 모델은 양성만을 반환하므로, 보조 필터에서 음성이 반환되어야 한다. 하지만, 보조 필터는 해시 함수 또는 학습 해시 함수를 사용하여 모든 원소에 대해서 입력을 수행하기 때문에 거짓 음성을 반환할 수 없다. 위의 모순에 의해서 확장 학습 블룸 필터는 거짓 음성이 발생하지 않는다는 것을 알 수 있다.

3.2 공간 분석

모델의 크기를 model, 전체 보조 필터의 크기를 m, $\lfloor(1-\alpha)*m\rfloor$ 크기의 보조 필터를 m_b, $\lceil\alpha*m\rceil$ 크기의 학습 해시 함수 공간을 m_L 이라고 가정한다. 추가적으로 P, P_b , P_L 은 각각 전체 보조 필터, 나눠진 보조 필터 그리고 학습 해시 함수에 대한 거짓 양성 확률이다. 확장 학습 블룸 필터가 학습 블룸 필터와 같거나 적은 공간을 사용하기 위해서는 다음의 조건을 만족해야 한다.

$$
\begin{aligned}
m_L + m_B + model &\le m + model \\
m_L + m_B &\le m \\
m_L + \frac{-nlnP_B}{(ln2)^2} &\le \frac{-nlnP}{(ln2)^2} \\
m_L + \frac{-nlnP_B}{(ln2)^2} &\le \frac{-nlnP_BP_L}{(ln2)^2} \\
m_L &\le \frac{-nlnP_L}{(ln2)^2}
\end{aligned}
$$

m 과 m_b 는 보조 필터로 이를 블룸 필터의 비트 벡터 크기를 구하는 공식으로 대체할 수 있다. 또한, 학습 해시 함수와 보조 필터는 함께 사용되어 원소를 판별하기 때문에 $P = P_B * P_L$ 공식을 사용하여 위의 수식을 정리할 수 있다. 결과적으로 학습 해시 함수에 사용되는 공간인 m_L 은 P_L 의 거짓 양성 확률을 가지는 블룸 필터보다 같거나 작아야 성능이 개선된다. 즉, 학습 해시 함수가 블룸 필터보다 적은 공간을 사용하면서 동일한 거짓 양성 비율을 가지게 된다면 확장 학습 블룸 필터는 학습 블룸 필터에 비해서 공간을 적게 사용 할 수 있다.

Algorithm 1 Insert of Extended LBF

```
1.  Function insert(x):
2.    if f(x) > threshold:
3.      pass
4.    else:
5.      for i = 1, ... , k:
6.        B_B[h_i(x)] = 1
7.      end
8.      B_L[f(x)] = 1
9.    end
10. end
```

Algorithm 2 Query of Extended LBF

```
1.  Function query(x):
2.    if f(x) > threshold:
3.      return True
4.    else:
5.      for i = 1, ... , k:
6.        if B_B[h_i(x)] == 0:
7.          return False
8.        end
9.      end
10.     if B_L[f(x)] == 0:
11.       return False
12.     end
13.   end
14.   return True
15. end
```

4. 구현 사항

다음은 확장 학습 블룸 필터를 구현하며 발생했던 문제와 해결책에 대해서 살펴본다.

4.1 모델 탐색

학습 해시 함수의 성능은 모델과 비트 벡터의 크기에 종속적이다. 모델도 여러 번의 학습을 통해서 최종적으로 가장 좋은 모델을 선택하듯이, 확장 학습 블룸 필터에서도 성능이 좋은 모델을 탐색하는 과정을 필요로 한다. 예를 들어, 비트 벡터의 크기 m 이 10,000 이라면 학습 해시 함수가 사용되는 공간을 1 비트씩 증가하면서 확장 학습 블룸 필터의 성능을 측정해볼 수 있다. 하지만,

확장 학습 블룸 필터를 1 비트씩 탐색하면 총 10,000 회의 탐색이 필요하며, 이는 시간과 비용이 많이 든다는 단점이 있다. 반면에 확장 학습 블룸 필터의 변수 α를 0.01 씩 증가하면서 성능을 측정한다면, 100 회 탐색으로 성능이 좋은 확장 학습 블룸 필터를 찾을 수 있다. 결과적으로 성능이 좋은 확장 학습 블룸 필터를 찾는데 드는 시간과 비용을 절약할 수 있다.

4.2 모델의 정밀도

모델의 정밀도 문제는 학습 해시 함수에 입력을 하였으나, 질의하였을 때는 입력된 원소에 대해서 인식하지 못하는 문제이다. 수학적으로는 [0, 1] 사이에 무수히 많은 실수가 있겠지만, 공학적으로는 [0, 1] 사이에 있는 수를 모두 표현할 수 없다. 따라서, 이를 32 비트 또는 64 비트 부동소수점으로 표현하며 [0, 1] 사이의 일부 숫자만 정확히 표현이 가능하다. 확장 학습 블룸 필터를 생성할 때는 인공신경망의 배치 크기를 늘려서 생성에 걸리는 시간을 줄일 수 있다. 허나, 확장 학습 블룸 필터가 생성된 후에는 질의가 오면 바로 응답하기 위해서 배치 크기를 1 로 사용한다. 생성과 질의를 할 때의 배치 크기 차이로 인해서 동일한 원소임에도 불구하고 인공신경망이 다른 출력값을 반환하는 경우가 있다. 즉, 부동소수점 오류로 인하여 학습 해시 함수가 결정론적 특성을 잃어버리는 경우가 발생할 수 있음을 보인다. 모델의 정밀도 문제는 인공신경망 내부에 사용되던 32 비트 부동소수점을 64 비트 부동소수점으로 바꾸면 해결할 수 있다.

4.3 모델 조정

학습 블룸 필터[4]를 제안한 T. Kraska 는 모델로 글자 임베딩[7] 32 차원, GRU[8] 16 차원 그리고 완전 연결 계층 1 차원을 사용할 것을 제안한다. 본 논문에서는 인공신경망의 구조를 글자 임베딩 16 차원, GRU 16 차원, 완전 연결 계층 8 차원 그리고 완전 연결 계층 1 차원으로 인공신경망 모델을 조정한다. 이를 통해서 기존의 4,017 개의 변수로 이루어진 인공신경망을 2,577 개로 줄임과 동시에 거짓 양성 비율에 대해서는 개선된 성능을 갖는 것을 보인다. 즉, 인공신경망에 사용되는 메모리를 40KB 에서 20KB 로 절약하고, 절약된 20KB 의 공간을 보조 필터에 사용하는 방법이라고 볼 수 있다.

5. 실험

실험은 Intel Core i5 4690 @ 3.50Ghz CPU 와 8GB Samsung DDR3 RAM 2 개를 사용하는 환경에서 진행되었다. 사용한 소프트웨어로는 Conda 4.10.1,

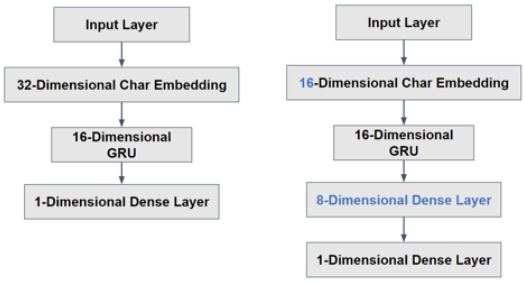

그림 2 모델 조정
Fig. 2 Model Adjustment

Python 3.7.4, Tensorflow 2.4 그리고 Scikit-learn 0.24.1 버전을 사용하였다. 위 실험 환경을 바탕으로 모델 탐색, 모델의 정밀도 그리고 모델 조정에 대한 실험 결과에 대해서 살펴본다.

5.1 데이터

인공신경망을 학습하기 위한 데이터는 Shalla's Blacklist[9]를 사용한다. Shalla's Blacklist 는 금융, 자동차 등의 81 개의 카테고리로 구성되어 있다. 각 카테고리별로 URL 정보가 존재하며 특정 카테고리의 접속을 방지하기 위한 데이터이다. 확장 학습 블룸 필터를 실험하기 위해서 성인용으로 구분된 URL 데이터 1,491,178 개와 그 외의 분야로 구분된 URL 데이터 1,435,527 개를 사용한다. 위 데이터를 학습에 사용할 때는 8:1:1 비율을 각각 학습, 개발, 테스트 데이터로 사용한다. 학습 데이터는 인공신경망을 학습하기 위해서 사용하고, 개발 데이터는 학습 블룸 필터의 임계치를 계산하기 위해서 사용된다. 마지막으로, 테스트 데이터는 모델의 성능을 평가할 때 사용된다.

5.2 모델

모든 실험은 본 논문에서 제안한 글자 임베딩 16 차원, GRU 16차원, 완전 연결 계층 8 차원 그리고 완전 연결 계층 1 차원으로 구성된 확장 학습 블룸 필터를 기준으로 한다. 위 인공신경망에 대한 입력 계층은 50 차원을 사용하고 이는 URL 50 글자를 읽어서 사용하는 것을 의미한다.

글자 임베딩은 글자를 벡터로 변환해주는 역할을 수행한다. 글자 임베딩은 glove.6B.50d 로 미리 학습된 단어 임베딩 데이터를 사용해서 만들 수 있다. 위의 glove.6B.50d 는 단어 임베딩에 60 억개의 단어를 50 차원의 벡터로 구성하였음을 의미한다. 단어 임베딩을 글자 임베딩으로 바꾸기 위해서는 각 글자가 사용되는 모든 단어 벡터에 평균을 취한다. 위의

연산으로 얻어진 글자 임베딩은 68 개의 글자에 대해서 50 차원의 벡터를 가진다. 단, 인공신경망은 16 차원의 글자 임베딩을 사용하므로 위에서 구한 50 차원의 글자 임베딩을 주성분 분석(PCA)을 통해서 16 차원으로 축소해서 사용한다.

GRU 는 자연어 처리를 위해서 사용된 계층이다. 다음으로 사용된 완전 연결 계층의 경우에는 모델의 정확성을 높이기 위해서 추가한 계층이며, ReLu 활성화 함수를 사용한다. 출력 계층은 [0, 1] 사이의 숫자를 출력하기 위해서 시그모이드(Sigmoid) 활성화 함수를 사용한다.

인공신경망의 손실 함수는 교차 엔트로피(Cross Entropy)를 사용한다. 이는 이진 분류를 할 때 사용되며, 블룸 필터가 해결하려고 하는 문제와 동일하다고 볼 수 있다. 인공신경망의 학습과 관련된 하이퍼파라미터는 학습률 0.005, 배치 크기 1024 그리고 에포크는 40 으로 설정한 상태로 실험을 진행한다. 단, 학습이 끝난 후에는 인공신경망의 배치 크기를 1 로 사용한다.

5.3 모델 탐색 실험

모델 탐색 실험은 확장 학습 블룸 필터의 α 변수를 0.01 에서 0.50 까지 변경하면서 거짓 양성 비율을 블룸 필터 그리고 학습 블룸 필터와 비교한다. 단, 전처리 과정에 사용되는 인공신경망은 모두 동일하며 데이터의 개수를 1,000 개, 10,000 개, 100,000 개, 1,000,000 개 사용한다.

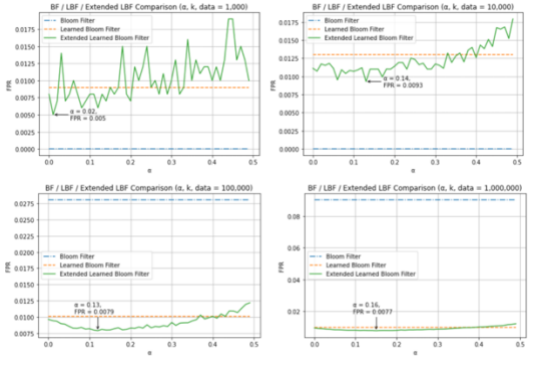

그림 3 모델 탐색 실험 결과

Fig. 3 Model Navigation Experimental Results

실험 결과를 보면 1,000 개와 10,000 개의 데이터가 사용된 경우에는 확장 학습 블룸 필터와 학습 블룸 필터는 블룸 필터에 비해서 좋지 않은 성능을 가지는 것을 살펴볼 수 있다. 이는 인공신경망의 크기가 수용해야 하는 데이터의 개수보다 크기 때문에 일어나는 현상이다. 또한, 학습 데이터가 많지 않아서 확장 학습 블룸 필터의 거짓 양성 비율 그래프에 노이즈가 발생하는 모습을 볼 수 있다.

반면에 데이터를 100,000 개와 1,000,000 개 사용하는 경우에는 확장 학습 블룸 필터와 학습 블룸 필터가 블룸 필터에 비해서 거짓 양성 비율이 압도적으로 좋은 것을 확인할 수 있다. 또한, 학습 데이터가 많아져서 확장 학습 블룸 필터의 거짓 양성 비율에 발생하는 노이즈가 대폭 줄어든 모습을 볼 수 있다. 즉, 학습 해시 함수를 사용하기 위해서는 일정량의 학습 데이터가 주어져야 한다는 것을 실험적으로 보인다. 결과적으로 α변수를 사용하여 선택된 확장 학습 블룸 필터는 학습 블룸 필터에 비해서 약 20% 개선된 거짓 양성 비율을 가진다.

5.4 모델의 정밀도 실험

모델의 정밀도 실험은 학습 해시 함수를 사용하는 환경에서 배치 크기를 조정하는 경우 거짓 음성 오류가 발생할 수 있음을 실험적으로 보인다. 모델의 정밀도 실험은 확장 학습 블룸 필터의 α 변수를 0.99 로 설정한 상태에서, 보조 필터의 크기를 강제적으로 10KB, 100KB, … , 10GB 의 공간을 사용하도록 설정한다. 또한, 32 와 64 비트 부동소수점 인공신경망을 100,000 개의 데이터로 학습시켜서 확장 학습 블룸 필터의 참 양성 비율을 확인한다.

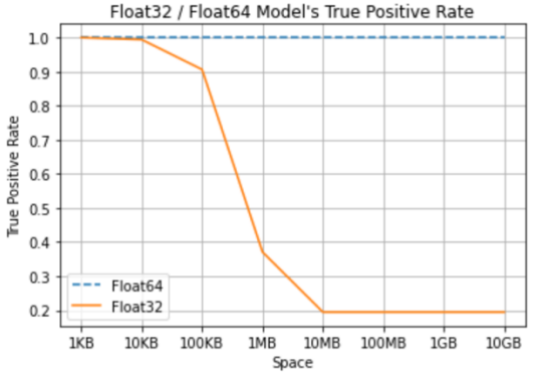

그림 4 모델의 정밀도 실험 결과

Fig. 4 Model Precision Experimental Results

실험 결과를 보면 32 비트 부동소수점 인공신경망을 사용하는 확장 학습 블룸 필터의 경우 10KB 부터 거짓 음성 오류가 발생하는 것을 확인할 수 있다. 반면에, 64 비트 부동소수점 인공신경망을 사용하는 확장 학습 블룸 필터의 경우에는 10GB 의 보조 필터 크기에서도 오류가 발생하지 않는 것을 확인할 수 있다. 따라서, 학습

해시 함수를 사용하는 공간이 10KB 이상인 경우에는 64 비트 부동소수점의 인공신경망을 사용해야 거짓 음성 오류가 발생하지 않는 것을 확인할 수 있다.

5.5 모델 조정 실험

모델 조정 실험은 2 번의 실험을 통해서 이루어진다. 첫번째 실험은 블룸 필터, 학습 블룸 필터 그리고 확장 학습 블룸 필터의 질의 시간을 비교한다. 실험은 각 블룸 필터에 대해서 질의를 10,000 회 수행하는 방식으로 진행된다. 질의에 사용된 데이터는 블룸 필터에 입력된 데이터와 입력되지 않은 데이터에 대해서 동일한 실험을 총 10 회씩 진행하고, 10 회에 대한 평균 값을 각 블룸 필터의 질의시간으로 사용하였다. T. Kraska 가 제안한 모델은 32 차원의 글자 임베딩을 사용하므로 접미사로 32 를 사용하며, 본 논문에서 제안한 모델은 16 차원의 글자 임베딩을 사용하므로 접미사로 16 을 사용한다.

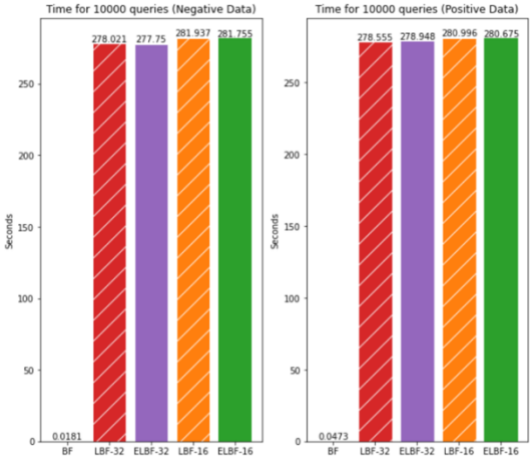

그림 5 모델 조정에 대한 시간 측정

Fig. 5 Time Measurement for Model Tuning

실험 결과를 보면 학습 블룸 필터와 확장 학습 블룸 필터의 질의시간은 유사한 것을 확인할 수 있다. 반면에, 본 논문에서 제안한 인공신경망의 구조가 기존의 T. Kraska 가 제안한 모델에 비해서 2 ~ 3 초 느린 것을 확인할 수 있다. 마지막으로 학습 계열의 블룸 필터는 블룸 필터에 비해서 10,000 배 정도 느린 성능을 갖는데, 이는 학습 계열 블룸 필터가 인공신경망을 사용하여 공간적으로 효율적인 대신에 갖는 약점이라고 볼 수 있다.

두번째 실험은 동일한 공간이 주어진 경우 블룸 필터, 학습 블룸 필터 그리고 확장 학습 블룸 필터의 거짓 양성 비율에 대해서 비교한다. 실험에 사용되는 공간은 30KB 부터 120KB 의 공간을 사용하였으며, 학습 계열 블룸 필터에 사용되는 인공신경망은 100,000 개의 데이터를 사용하여 학습하였다.

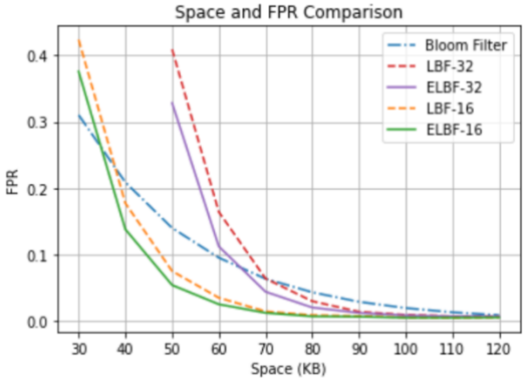

그림 6 모델 조정에 대한 거짓 양성 비율

Fig. 6 False Positive Rate for Model Tuning

실험 결과를 보면, 30KB 에서는 여전히 블룸 필터가 가장 좋은 거짓 양성 비율을 갖는다. 허나, 40KB 부터는 본 논문에서 제안한 인공신경망을 사용하는 확장 학습 블룸 필터가 가장 좋은 거짓 양성 비율을 가지는 것을 확인할 수 있다. 추가적으로 T. Kraska 가 제안한 인공신경망을 사용했을 때는 인공신경망의 크기로 인해서 40KB 이전에서는 측정이 불가능하였으며, 이는 본 논문에서 제안한 확장 학습 블룸 필터에 비해서 좋지 않은 거짓 양성 비율을 가진다.

6. 결론

본 논문은 학습 블룸 필터의 변형인 확장 학습 블룸 필터를 제안한다. 확장 학습 블룸 필터는 보조 필터의 공간 일부를 학습 해시 함수로 사용하도록 하며, 학습 해시 함수가 사용되는 공간은 변수 α를 통해서 결정한다. 실험을 통해서 학습 블룸 필터와 동일한 공간을 사용하는 확장 학습 블룸 필터가 개선된 거짓 양성 비율과 유사한 질의 시간을 갖는 것을 실험을 통해서 확인하였다. 또한, 학습 해시 함수를 사용함으로 발생할 수 있는 모델의 정밀도 문제에 대해서 소개하며, 이를 64 비트 부동소수점을 사용하여 해결할 것을 제안한다. 마지막으로, 인공신경망의 성능과 확장 학습 블룸 필터의 성능이 꼭 비례하지 않음을 보였다. 인공신경망에 사용되는 공간을 보조 필터에 사용하면, 확장 학습 블룸 필터 관점에서는 개선된 거짓 양성 비율을 가질 수 있음을 실험을 통해서 확인하였다.

확장 학습 블룸 필터는 전처리 과정에 사용되는 인공신경망에 의해서 질의 시간이 느리다는 단점이 있다. 인공신경망 외의 다른 경량화된 모델을 적용한다면 질의 시간이 개선될 수 있을 것이라고 판단된다. 또한, 학습 해시 함수에서 모델의 출력값 f(x)에 비트 벡터의 크기 m 을 곱하는 것은 공간을 균일하게 사용한다는 것을 의미한다. 모델의 출력값 f(x) 중에 오류가 많이 발생하는 구간에 가중치를 주는 수식을 추가할 수 있다면, 학습 해시 함수의 거짓 양성 비율이 개선될 수 있을 것이라고 판단된다.

References

[1] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, Vol. 13, No. 7, pp. 422–426, Jul. 1970.

[2] N. Dayan, M. Athanassoulis, and S. Idreos. Monkey: Optimal Navigable Key-Value Store. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 79–94, May. 2017.

[3] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers", *In IEEE Symposium on Security and Privacy*, June. 2017.

[4] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," *In International Conference on Management of Data (ACM SIGMOD)*, pp. 489-504, Jun. 2018.

[5] A. Bhattacharya, B. Srikanta and B. Amitabha, "Adaptive Learned Bloom Filters under Incremental Workloads," *In Proc. of the 7th ACM IKDD CoDS and 25th COMAD*, pp. 107-115, Jan. 2020.

[6] K. Vaidya, E. Knorr and T. Kraska, "Partitioned Learned Bloom Filter," arXiv preprint arXiv:2006.03176, 2020.

[7] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", *In Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP)*, pp. 1724–1734, Oct. 2014.

[8] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global vectors for word representation." *In Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532– 1543, Jan. 2014.

[9] Shalla Secure Services KG [Online]. Available: https://www.shallalist.de (Accessed: 5 July 2021)