대용량 데이터에 대한 효율적인 L-diversity 비식별화 구현

전민혁*, Odsuren Temuujin*, 안진현**, 임동혁*
*호서대학교 컴퓨터공학과
*e-mail : jeoncoder@gmail.com, temuujintemka@gmail.com, dhim@hoseo.edu
**제주대학교 경영정보학과
**e-mail : jha@jejunu.ac.kr

Implementation of efficient L-diversity de-identification for large data

Min-Hyuk Jeon*, Odsuren Temuujin*, Jinhyun Ahn**, Dong-Hyuk Im*
*Dept. of Computer Engineering, Ho-Seo University
**Dept. of Management Information Systems, Jeju National University

요 약

최근 많은 단체나 기업에서 다양하고 방대한 데이터를 요구로 하고, 그에 따라서 국가 공공데이터나 데이터 브로커등 데이터를 통해 직접 수집 하거나 구매해야 하는 경우가 많아지고 있다. 하지만 개인정보의 경우 개인의 동의 없이는 타인에게 양도가 불가능하여 이러한 데이터에 대한 연구에 어려움이 있다. 그래서 특정 개인을 추론할 수 없도록 하는 비식별 처리 기술이 연구되고 있다. 이러한 비식별화의 정도는 모델로 나타낼 수가 있는데, 현재 k-anonymity 와 l-diversity 모델 등이 많이 사용된다. 이 중에서 l-diversity 는 k-anonymity 의 만족 조건을 포함하고 있어 비식별화의 정도가 더욱 강하다. 이러한 l-diversity 모델을 만족하는 알고리즘은 The Hardness and Approximation, Anatomy 등이 있는데 본 논문에서는 일반화 과정을 거치지 않아 유용성이 높은 Anatomy 의 구현에 대해 연구하였다. 또한 비식별화 과정은 전체 데이터에 대한 특성을 고려해야 하기 때문에 데이터의 크기가 커짐에 따라 실질적인 처리량이 방대해지는데, 이러한 문제를 Spark 를 통해 데이터가 커짐에 따라서 최대한 안정적으로 대응하여 처리할 수 있는 시스템을 구현하였다.

1. 서론

단체나 기업에서 필요로 하는 데이터는 일반적으로 단시간 내에 충분히 확보하기가 어렵다. 그렇기에 대다수는 정부에서 제공하는 공공데이터나 데이터 구매를 통해 데이터를 얻는다. 하지만 개인정보가 포함된 데이터는 개개인의 동의 없이 타인에게 제공할 수 없어 필요로 함에도 얻을 수 없는 제한이 있다.

이러한 문제를 해결하기 위한 연구로 비식별화 처리가 연구되고 있는데, 비식별화란 원본 데이터에 대해서 일부나 전체를 삭제하여 다른 정보들과 결합하더라도 특정 개인과 연결할 수 없도록 하는 기술이다. 비식별화 된 데이터는 개인정보의 특성을 갖고있지 않아 배포가 가능하고, 이러한 비식별화에는 비식별 정도를 나타내는 모델이라는 척도가 존재한다. 현재까지 연구된 모델은 k-anonymity[1], l-diversity[2] 등이 있다. l-diversity 모델은 k-익명성에 비해 비식별화 정도가 더욱 강화된 모델인데, l-diversity 의 경우 The Hardness and Approximation[3], Anatomy[4] 등의 알고리즘이 연구되었다. 이러한 두 개의 알고리즘 중 데이터의 일반화 과정을 거치지 않는 Anatomy 알고리즘을 통해 정보의 소실을 최소화하는 더욱 유용한 비식별화 구현을 하였다. 그리고 대용량 데이터의 처리에 있어서 단일 서버로는 대용량 처리 불가, 처리 속도 저하 등의 문제가 발생하여 Spark 분산처리 플렛폼[5]을 이용 여러 서버로 분할처리하여 처리 효율을 높였다.

2. 관련연구

비식별화에 사용되는 공통적인 개념은 다음과 같다. 우선 비식별화 할 데이터는 각각의 속성별로 나뉘어 있어야 한다. 그리고 데이터의 특정 속성으로 데이터 전체를 식별할 수 있는 속성을 식별자 (ID; Identifier) 라고 정의한다. 또한 단일 속성만으로는 특정 데이터를 식별할 수 없지만 다른 속성들과의 결합 또는 추가적인 외부 데이터를 통해 식별할 수 있을 가능성을 가지고 있는 속성은 준식별자 (QI; Quasi-Identifier) 라고 정의한다. 그리고 데이터에서 노출되면 안되는 가장 중요하거나 민감한 속성값을 민감속성 (SA; Sensitive Attribute) 라고 정의한다. 그리고 최종 비식별 데이터에서 동일한 준식별자를 가지고 있는 데이터

또는 하나의 그룹으로 묶인 데이터들을 동질클래스 (EC; Equivalence Class) 로 정의한다.

Name	Research Interest	Doctoral DegreeFrom	Masters DegreeFrom	Undergraduate DegreeFrom
Professor7	Research28	University45	University27	University7
Professor4	Research2	University11	University11	University5
Professor5	Research33	University35	University12	University14
Professor2	Research15	University17	University6	University14
Professor9	Research10	University2	University15	University25

(그림 1) 테이블로 가공된 LUBM 원본 데이터

Group Number	Research Interest	Doctoral DegreeFrom	Masters DegreeFrom
1	Research28	University45	University27
1	Research2	University11	University11
2	Research33	University35	University12
2	Research15	University17	University6
2	Research10	University2	University15

QIT

Group Number	Undergraduate DegreeFrom	Frequency
1	University7	1
1	University5	1
2	University14	1
2	University14	1
2	University25	1

ST

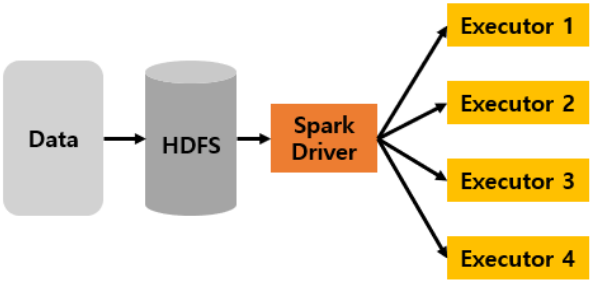
(그림 2) 2-diversity Anatomy 비식별화 결과

Anatomy 알고리즘은 l-diversity 모델을 만족하는 모델로써, 구현 방법은 다음과 같다. 우선 (그림 1) 과 같은 속성값을 가지고 있는 데이터에서 ID 속성값을 제거한다. 그리고 SA 를 기준으로 데이터를 그룹화한 SA Bucket 을 만들고, 각 Bucket 에서 1 개씩의 데이터를 추출하여 총 1 개가 되면 하나의 그룹으로 만들고, 이러한 과정을 모든 Bucket 이 없을 때까지 반복하면 그룹이 만들어진다. 그리고 (그림 2)와 같이 그룹을 QI, 그룹번호를 속성값으로 갖는 QIT(QI Table)와 그룹번호, 민감속성, 민감속성의 개수를 속성값으로 갖는 ST(SA Table) 로 나뉘어 비식별 처리를 완료한다. 또한 각각의 그룹을 하나의 EC 로 표현하는데, 각 EC 는 고유한 SA 를 최소 1 개 가지고 있어 특정 데이터를 최소 l-1 개의 데이터와 구분이 불가능하여 l- diversity 을 만족한다.

3. 시스템 구현

이러한 시스템의 구조는 다음 (그림 3) 과 같다. 우선 데이터를 일련의 과정을 통해 HDFS(Hadoop Distributed File System)에 속성값을 가지는 테이블 형태의 분산파일로 저장된다. 그리고 HDFS 파일의 로드와 동시에 ID, QI, SA 를 지정하여 SA 는 제거 한 뒤에 RDD(Resilient Disributed Data) 라는 최소단위인 Task 로 분할한다. 그리고 각각의 Task 는 Driver 는 각각의 Executor 로 할당된다. 이후 Anatomy 의 알고리즘을 따라 RDD 를 Mapping 하여 Bucket 을 만든다. 그리고 1 개씩의 데이터를 Filtering 하여 새로운 RDD 들로 Map 그룹을 만들고, 최종적으로 이러한 그룹들은 Key 값인 그룹 번호로 GroupBy 되어 QIT 와 ST 의 EC 가 된다. 이후 나뉘어진 RDD 들은 합쳐져 QIT 와 ST 각각 HDFS 에 저장된다.

이러한 시스템은 최대로 처리할 수 있는 파일 용량의 한계선은 있지만, 노드를 추가함에 따라 단일 서버 시스템에 비해 효율적인 확장을 통해 빠른 처리속도로 비식별화 데이터 제공이 가능하다.



(그림 3) 시스템 구조

4. 실험 결과

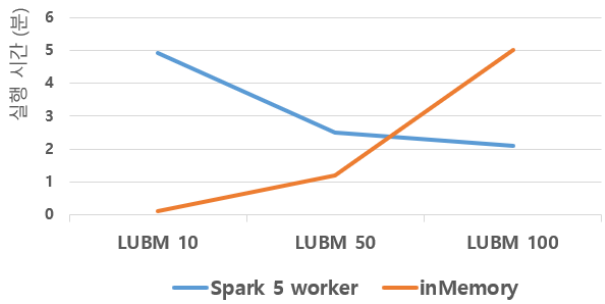
4.1 실험 환경

본 논문에서의 실험은 LUBM-(10, 50, 100, 700)[6] 데이터를 사용하였다. 이 데이터는 벤치마킹용 데이터로서 10, 50, 100, 700 개의 학교 내의 교직원에 대한 개인정보 데이터이다. 또한 Spark 의 구성 노드는 총 5 개로, 각 노드별 CPU 는 4 Core, 메모리는 20Gb 이고, 총 5 개의 Worker 노드를 1 개의 Driver 노드와 최대 4 개의 Executor 노드로 나누어 실험하였다.

4.1 성능 분석

우리는 데이터의 비교를 위하여 두가지 시스템을 구현하였는데, Spark 를 사용하지 않은 단일서버 InMemory 상에서의 Anatomy 비식별 시스템과 Spark 를 사용하여 유동적으로 추가 서버를 연결할 수 있는 시스템이다.

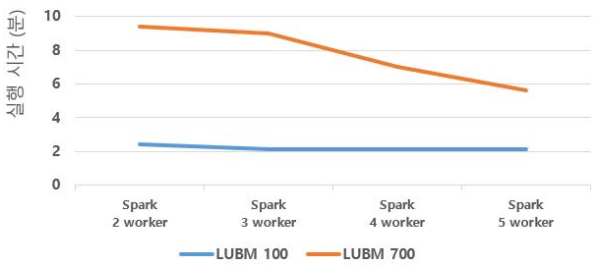
우선 (그림 4) 에서는 LUBM 100 이하의 저용량 데이터를 테스트하였는데, 이러한 저용량에서는 Spark 의 Partition 분할 처리방식이 오히려 성능 저하의 원인이 되어 느려짐을 볼 수 있다. 그리고 InMemory 의 경우 작은 데이터에서는 Spark 시스템에 비해 상대적으로 유리하지만, 데이터의 크기가 증가함에 따라서 실행시간이 급격히 증가하였다. 그에 반해 Spark 시스템은 데이터의 크기가 늘어남에 따라 데이터크기에 비해 실행시간이 줄어들었다.



(그림 4) 두 시스템간의 비교

이후로 Spark 가 InMemory 에 비해 큰 데이터에 유리한 실험 결과를 바탕으로 데이터의 크기에 따른 Spark 의 Worker 수로 비교를 해보았고, 그에 따른 결과 그래프는 (그림 5) 이다. 같은 데이터라고 하더라도 Worker 수의 증가에 따라 실행시간은 꾸준히 감소

하고 있는 모습을 볼 수 있는데, 이러한 감소폭은 데이터의 크기가 커질수록 더욱 커짐을 확인할 수 있다. 이는 Driver 노드가 각각의 Executor 노드에 Task 를 분할 할당하는 시간은 절대적이고, Executor 가 증가하면 Task 처리 효율이 증가함에 따라 발생하는 현상이다.



(그림 5) Spark Worker 노드 수 별 실행시간 비교

우리는 InMemory 단일 시스템과 Spark 시스템을 구현해보았고, 위와 같은 실험 테스트를 통해서 Spark 를 사용한 시스템이 InMemory 에 비해 큰 데이터에 대해 더욱 향상된 처리 효율을 보여줌을 확인할 수 있었다.

5. 결론

본 논문에서는 대용량 데이터에 대해 l-diversity 비식별화 모델을 적용하는 시스템을 구현하였다. 단일 서버에서는 처리가 불가능하거나 오래걸리는 데이터에 대하여 Spark 를 적용한 시스템은 저비용으로 추가적인 확장이 가능하고, Anatomy 비식별화의 최종 결과물은 원본과 비슷해 데이터로서의 유용성도 크다. 하지만 l-diversity 도 완전히 안전한 비식별화 모델은 아니다. QIT 의 EC 의 QI 분포간의 차이가 커지게 되면 추론이 가능하기 때문이다. 이러한 단점을 보완하기 위해 t-closeness 모델도 연구되고 있는데, 대용량 데이터에 대한 보다 비식별성이 강화된 t-closeness 모델의 연구가 필요하다. 또한 본 논문에서 진행한 l-diversity 연구는 고정된 데이터에 대한 처리를 다루었는데, 추가적으로 데이터의 추가 또는 제거에 따른 유동적인 비식별처리 시스템의 구현[7]에 대해서도 연구를 진행하고 있다.

Acknowledgement

이 논문은 2017 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구 (No.NRF-2017R1C1B1003600)이며, 2018 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2018R1D1A1B07048380). 또한, 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음 (IITP-2019-2018-0-01417).

참고문헌

[1] SWEENEY, Latanya. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, 10.05: 557-570.

[2] Machanavajjhala, Ashwin, et al.: l-diversity: Privacy beyond k-anonymity. In: 22nd Interna-tional Conference on Data Engineering (ICDE'06). IEEE, 2006. p. 24-24.

[3] XIAO, Xiaokui; YI, Ke; TAO, Yufei. The hardness and approximation algorithms for l-diversity. In: Proceedings of the 13th International Conference on Extending Database Technology. ACM, 2010. p. 135-146.

[4] XIAO, Xiaokui; TAO, Yufei. Anatomy: Simple and effective privacy preservation. In: Pro-ceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006. p. 139-150.

[5] ZAHARIA, Matei, et al. Apache spark: a unified engine for big data processing. Communi-cations of the ACM, 2016, 59.11: 56-65.

[6] GitHub. (2019). rvesse/lubm-uba. [online] Available at: https://github.com/rvesse/lubm-uba [Accessed 22 Sep. 2019].

[7] TEMUUJIN, Odsuren; AHN, Jinhyun; IM, Dong-Hyuk. Efficient L-diversity Algorithm for Preserving Privacy of Dynamically Published Datasets. IEEE Access, 2019.