프로그램 합성을 사용한 테이블 데이터에 대한 데이터 랭글링 프로그램의 학습

김유리 [1], 서인 [1], 한욱신 [1, 2]
포항공과대학교 창의 IT 융합공학과 [1]
포항공과대학교 컴퓨터공학과 [2]
e-mail : {yrkim, iseo, wshan}@dblab.postech.ac.kr

Learning of Data Wrangling Program for Table Data Using Program Synthesis

Yurie Kim[1], In Seo[1], and Wook-Shin Han[1, 2, *]
Dept. of Creative IT Engineering, Pohang University of Science and Technology (POSTECH) [1]
Dept. of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH) [2]

요 약

데이터 랭글링은 원시 데이터를 분석하기에 더 적합한 형태로 변환하는 프로세스를 말한다. 본 논문에서는 프로그램 합성 기술을 이용하여 테이블 데이터에 대하여 사용자의 의도를 만족하는 데이터 랭글링 프로그램을 자동 생성하는 방법을 제안한다. 제안하는 방법은 입/출력 테이블 예시를 명세로 받아 연산자 시퀀스를 탐색한다. 실험을 통해 제안하는 방법이 빠른 시간 안에 정확한 데이터 랭글링 프로그램을 학습할 수 있음을 보였다.

1. 서론

데이터 랭글링(data wrangling)이란 원시(raw) 데이터를 분석하기에 더 적합한 형태로 변환하는 프로세스로 다양한 데이터 분석에 앞서 반드시 수행되어야 하는 작업이다. 그러나 데이터 랭글링을 수작업으로 수행하려면 매우 많은 인력과 시간이 필요하며, 거대한 데이터를 다루기에 적합하지 않다. 데이터 변환을 위해 프로그램을 작성해 사용하면 입력-출력 형식에 따라 매번 새로운 프로그램을 작성해야 하는 비용이 필요하다. 클라우드플라워(CrowdFlower) 조사에 따르면, 데이터 과학자들은 80%의 시간을 데이터를 수집하고 랭글링 하는 데에 사용한다[1].

이러한 어려움을 개선하기 위하여 다양한 상업용, 학술용 데이터 랭글링 툴이 제안되었지만, 이들은 비전문가가 사용하기 어렵거나 제한적인 변환만을 지원하는 한계가 있다. 대표적인 상업용 데이터 랭글링 툴인 트리팩타(Trifacta)는 사용자와 상호작용을 통해 알맞은 테이블 변환을 찾아낸다. 그러나 테이블 변환에 대한 지식이 부족한 일반인은 트리팩터를 사용하기가 어렵다. 또 다른 데이터 랭글링 툴인 플래시릴레이트(FlashRelate)는 테이블에서 셀의 위치를 바꾸는 변환(layout transformation)은 제공하지만, 셀의 내용을 바꾸는 변환(syntactic transformation)은 제공하지 않는다 [2].

프로그램 합성(program synthesis)은 주어진 명세(specification)를 만족하는 프로그램을 자동으로 생성하는 기법이다. 사용자는 생성된 프로그램을 이용하여 임의의 데이터에 대해 원하는 컴퓨팅 결과를 얻을 수 있다[3].

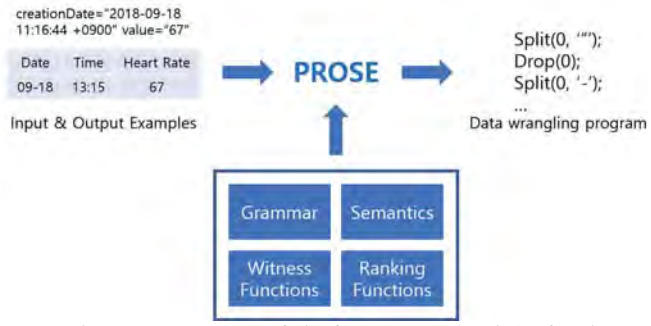
본 논문에서는 프로그램 합성 기술을 이용하여 테이블 데이터에 대한 데이터 랭글링 프로그램을 학습하는 방법을 제안한다. 제안하는 방법은 대상 테이블 T_{raw}의 일부인 입력 테이블 T_i를 입력 예시로, T_i을 데이터 랭글링한 결과인 출력 테이블 T_o를 출력 예시로 하는 예제들을 명세로 받아 데이터 랭글링 프로그램을 학습한다. 제안하는 방법을 이용하면 비전문가가 테이블 변환의 중간 과정을 모르더라도 입/출력 예시로부터 자신의 의도에 맞는 프로그램을 학습할 수 있고, 학습된 프로그램을 사용하여 전체 테이블 T_{raw} 을 데이터 랭글링 할 수 있다. 제안하는 방법을 이용하면 T_{raw} 의 크기가 크더라도 사용자는 테이블 일부에 대해서만 직접 데이터 랭글링을 수행하므로, 데이터 랭글링에 필요한 시간을 크게 절약할 수 있다.

2. 프로즈(PROSE) 프레임워크

본 논문에서 제안하는 데이터 랭글링 프로그램의 학습 방법은 마이크로소프트에서 개발한 프로그램 합성 프레임워크인 프로즈[4]를 이용하여 구현하였다.

* 교신 저자

그림 1 은 프로즈 프레임워크에서 프로그램을 학습하는 예를 보여준다. 프로즈를 이용해 프로그램을 합성하기 위해, 개발자는 문법(grammar), 시맨틱스(semantics), 증인 함수(witness functions), 그리고 랭킹 함수(ranking functions)를 구현해야 한다. 문법은 합성할 프로그램의 입/출력 자료형과 프로그램 공간(space)을 정의하고, 시맨틱스는 문법을 이루는 연산자(operator)들이 어떻게 동작하는지를 정의한다. 증인 함수는 문법으로부터 정의되는 프로그램 공간 내에서 후보 프로그램을 찾기 위해 사용하는 탐색 방법을 정의하며, 랭킹함수는 탐색의 결과로 얻어진 후보 프로그램들의 순위를 매기는 데 사용된다. 정의된 문법, 시맨틱스, 증인 함수와 랭킹함수가 주어지면, 사용자는 입/출력 예시들만을 이용해 원하는 프로그램 또는 순위가 매겨진 프로그램들을 얻어낼 수 있다.



<그림 1> 프로즈를 이용한 프로그램 학습의 예.

3. 데이터 랭글링 프로그램의 학습

본 논문에서는 프로즈를 이용하여 프로그램 합성 기반의 데이터 랭글링 프로그램을 학습하기 위해 목적에 맞는 문법, 시맨틱스, 증인 함수, 그리고 랭킹 함수를 구현하였다.

3.1 문법과 시맨틱스의 구현

문법에서는 프로그램의 입/출력 자료형과 프로그램 공간을 정의한다. 먼저 입/출력 자료형은 테이블이다. 문법에서 정의하는 프로그램 공간은 변환검색(TransformSearch)이라는 단일 연산자로 이루어진 프로그램을 학습하도록 설계하였다. 변환검색 연산자는 입력 테이블과 연산자 시퀀스를 입력으로 받아 입력 테이블에 연산자를 순차적으로 적용한 결과를 출력하는 연산자이다. 시맨틱스에는 <표 1>의 테이블 변환 연산자가 정의되어 있다.

<표 1> 테이블 변환 연산자의 종류와 동작.

Merge	두 개의 열을 합친다.
Split	한 열을 구분 문자(delimiter)로 나눈다.
Fill	빈 셀을 같은 열의 셀 값으로 채운다.
Drop	한 열을 삭제한다.
Move	한 열을 오른쪽 끝으로 옮긴다.
Fold	테이블을 1-디맨션(dimension)으로 줄인다.
Unfold	Fold 의 역 연산자이다.
Lookup	다른 테이블에서 값을 가져온다.

3.2 증인 함수와 랭킹 함수의 구현

증인 함수에서는 변환검색 연산자의 입력인 연산자 시퀀스를 탐색한다. 입력 테이블 T_i 를 출력 테이블 T_o 로 변환할 수 있는 연산자 시퀀스는 무한히 존재하기 때문에, 한 테이블을 다른 테이블로 변환시키는 데 필요한 테이블 변환 연산자의 수를 근사하는 TED(Table Edit Distance) 함수를 사용한다 [5]. TED 값으로 연산자 시퀀스를 구성하기 때문에 랭킹 함수의 역할도 같이 수행하게 된다

연산자 시퀀스 탐색은 먼저 T_i 에 가능한 연산자들의 집합 O 를 만든다. 연산자에 매개변수로 열이 있으면, 테이블의 각 열에 적용되는 연산자를 O 에 추가한다. 연산자에 매개변수로 구분 문자가 있으면, 테이블에 포함된 모든 구분 문자의 집합에서 각 구분 문자를 매개변수로 갖는 연산자를 O 에 추가한다. T_i 에 O 의 연산자를 한 개씩 적용한 후보 테이블 집합을 만든다. 후보 테이블 집합에서 T_o 과 TED 값이 가장 작은 후보 테이블이 첫 번째 중간 테이블 T_{inter1} 이 되고, 이 테이블에 적용된 연산자를 연산자 시퀀스에 추가한다. T_{inter1} 에 대해 같은 과정을 반복하여 두 번째 중간 테이블을 찾고, 중간 테이블이 T_o 일 때까지 이 과정을 반복한다.

4. 실험

본 절에서는 제안하는 데이터 랭글링 프로그램 학습 방법을 평가하기 위해 수행한 테스트 케이스를 소개하고 그 결과를 분석한다.

4.1 실험 환경

실험에는 날짜, 시간, 심박수로 구성된 심박수 데이터와 저자, 저널명, 권(volume), 연도, 페이지 수로 구성된 논문 데이터를 직접 생성하여 사용하였다.

심박수 데이터에 대한 실험의 경우, <표 2>의 날짜, 시간, 심박수로 이루어진 입력 예시 테이블을 <표 3>과 같은 출력 예시 테이블로 변환하는 프로그램을 학습한다. 심박수 데이터의 입/출력 테이블은 총 72 개의 행으로 이루어져 있으며, 이 중 첫 2 개의 행과 4 개의 행을 예시로 이용했을 때의 학습을 실험하였다.

논문 데이터에 대한 실험의 경우, <표 4>와 같이 정리되지 않은 입력 예시 테이블을 <표 5>와 같이 저자, 저널명, 권, 페이지, 연도로 열이 구분된 출력 예시 테이블로 변환하는 프로그램을 학습한다. 논문 데이터의 입/출력 테이블은 총 97 개의 행으로 이루어져 있으며, 이 중 1 개의 행과 2 개의 행을 이용했을 때의 학습을 실험하였다.

<표 2> 심박수 데이터 테이블의 입력 예시.

날짜	시간	심박수
2018-09-19	11:31	67
	11:32	70
2018-09-20	11:35	68
	11:36	65

<표 3> 심박수 데이터 테이블의 출력 예시.

월	일	시	분	심박수
09	19	11	31	67
09	19	11	32	70

<표 4> 논문 데이터 테이블의 입력 예시.

논문
J.C.Summers, S.A.Ausen, J.Catal. 58 (1979) 131~143
J.C.Schlatter, P.J.Mitchell, Ind.Eng.Prod.Res.Dev. 19 (1980) 288~293

<표 5> 논문 데이터 테이블의 출력 예시.

저자	저널명	권	페이지	연도
J.C.Summers,S.A.Ausen	J.Catal.	58	131~143	(1979)
J.C.Schlatter,P.J.Mitchell	Ind.Eng.Prod. Res.Dev.	19	288~293	(1980)

모든 실험은 Intel Core i7-3517U CPU 2.40 GHz, 4GB RAM, Windows 10 Pro OS 환경에서 진행하였고, PROSE 6.7 버전을 사용하였다.

4.2 실험 결과

각 실험에서는 제안하는 데이터 랭글링 프로그램 학습 방법의 학습 정확도와 학습에 걸리는 시간을 평가한다.

<표 6>은 각 실험에서 학습된 결과 연산자 시퀀스를 보여준다. 이 연산자 시퀀스들을 학습에 사용되지 않았던 전체 입력 테이블에 적용한 결과를 전체 출력 테이블과 비교함으로써 학습의 정확도를 평가하였다. 모든 실험에서 제안하는 학습 방법은 학습에 사용했던 행의 수와 관계없이 정확한 결과를 얻어내는 데이터 랭글링 프로그램을 얻어내었다.

<표 7>은 각 실험에서 데이터 랭글링 프로그램을 학습하는데 소요된 시간을 보여준다. 각 실험은 총 3회 수행하여 학습 시간을 측정하였으며, 두 데이터셋에서 모두 학습에 사용된 행의 개수가 늘어날수록 더 학습 시간이 길어졌다. 이는 연산자 시퀀스를 탐색할 때 사용하는 TED 함수의 시간 복잡도가 테이블의 셀 개수에 비례하기 때문이다. 또한, 테이블에 포함된 구분 문자가 많을수록, 후보 연산자 집합의 크기가 커지기 때문에, 심박수 데이터 실험보다 논문 데이터 실험에서 학습에 더 오랜 시간이 걸렸다.

<표 6> 연산자 시퀀스 학습 결과.

	심박수 데이터	논문 데이터

연산자 시퀀스	split(1, ":");
	fill(0);
	split(0, "-");
	drop(0);
		split(0, " ");
		split(1, ",");
		merge(0, " ");
		move(3);

<표 7> 학습 시간 측정 결과.

	1 회(ms)	2 회(ms)	3 회(ms)	평균(ms)
심박수 (2 행)	19,958	19,629	19,391	19,659
심박수 (4 행)	73,208	77,218	73,795	74,740
논문 (1 행)	8,700	8,707	8,573	8,660
논문 (2 행)	31,576	32,988	31,258	31,941

5. 결론

본 논문에서는 프로그램 합성에 기반을 둔 입/출력 예시로부터 테이블 데이터에 대한 데이터 랭글링 프로그램의 학습 방법을 제안하였다. 제안하는 방법을 이용하면 입/출력 예시로부터 데이터 과학 비전문가도 빠른 시간 안에 데이터 랭글링 프로그램을 생성할 수 있다. 또한, 실제 사용 예시와 가까운 실험을 통해 제안하는 방법이 빠른 시간 안에 정확한 데이터 랭글링 프로그램을 학습할 수 있음을 보였다. 현재 연구는 테이블 형태의 입/출력 데이터의 형식이 테이블로 고정되어 있으나, 더 다양한 응용을 위해서는 텍스트 데이터, XML 데이터와 같이 더 다양한 형식의 데이터를 처리할 수 있도록 확장하는 연구가 필요하다. 더불어 제안하는 방법의 효용성을 입증하기 위해 향후에는 더 크고 다양한 데이터셋들을 이용한 실험을 통해 성능을 평가하고 알고리즘을 개선하는 연구를 수행하고자 한다.

6. 사사

이 논문은 2017 년도 정부 (과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업 (No.NRF- 2017M3C4A7065887)의 지원과 과학기술정보통신부 및 정보통신기획평가원의 ICT 명품인재양성사업(IITP-2019-2011-1-00783)의 연구결과로 수행되었음

참고문헌

[1] CrowdFlower, “2016 Data Science Report” (available at https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf).

[2] D. W. Barowy et al. "FlashRelate: extracting relational data from semi-structured spreadsheets using examples," In ACM, 2015.

[3] S. Gulwani, O. Polozov, and R. Singh, “Program Synthesis,” Foundations and Trends® in Programming Languages, 4: 1-119.

[4] Microsoft Prose, “Microsoft Program Synthesis using Examples SDK” (available at https://microsoft.github.io/prose/).

[5] Z. Jin et al., “Foofah: Transforming Data By Example,” In SIGMOD, 2017.