Extracting N-ary Facts from Wikipedia Table Clusters

Benno Kruit, Peter Boncz, Jacopo Urbani

Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020

서울과학기술대학교 데이터사이언스학과
20510082 전우진

Contents

1. Introduction

- 위키피디아에 존재하는 테이블들은 지식의 원천임
- 웹에서 가장 큰 지식 저장소는 그래프와 같은 Knowledge Base(KB) 형태임
  - 특히 위키피디아의 infobox를 바탕으로 만들어진 KB가 인기있음
  - 이 때, 테이블은 infobox에 포함된 지식을 보완하는데 쓰일 수 있음
  - 따라서 테이블은 현재 위키피디아 기반 KB를 확장하는데 중요한 원천이 됨

- 하지만 위키피디아의 테이블의 처리 자동화를 위해선 일관된 구조로 변환해야함
- 일관된 구조로의 변환은 다양한 이유로 어려움
  - 테이블의 다양한 레이아웃
  - N-ary relation

- 본 연구는 이런 어려움들을 부분적으로 해결하던 연구들과 달리 한번에 해결할 수 있는 새로운 방법을 제안함
  - 1. 서로 다른 레이아웃을 가진 테이블의 holistic schema 정규화를 위해 테이블 말뭉치 통계를 결합함
  - 2. 각각의 테이블들을 클러스터링을 통해 큰 테이블로 합침
  - 3. KB를 보완할 수 있는 정보를 위해 클러스터 테이블에서 entity-attribute와 n-ary 정보를 추출함

2. Our Approach

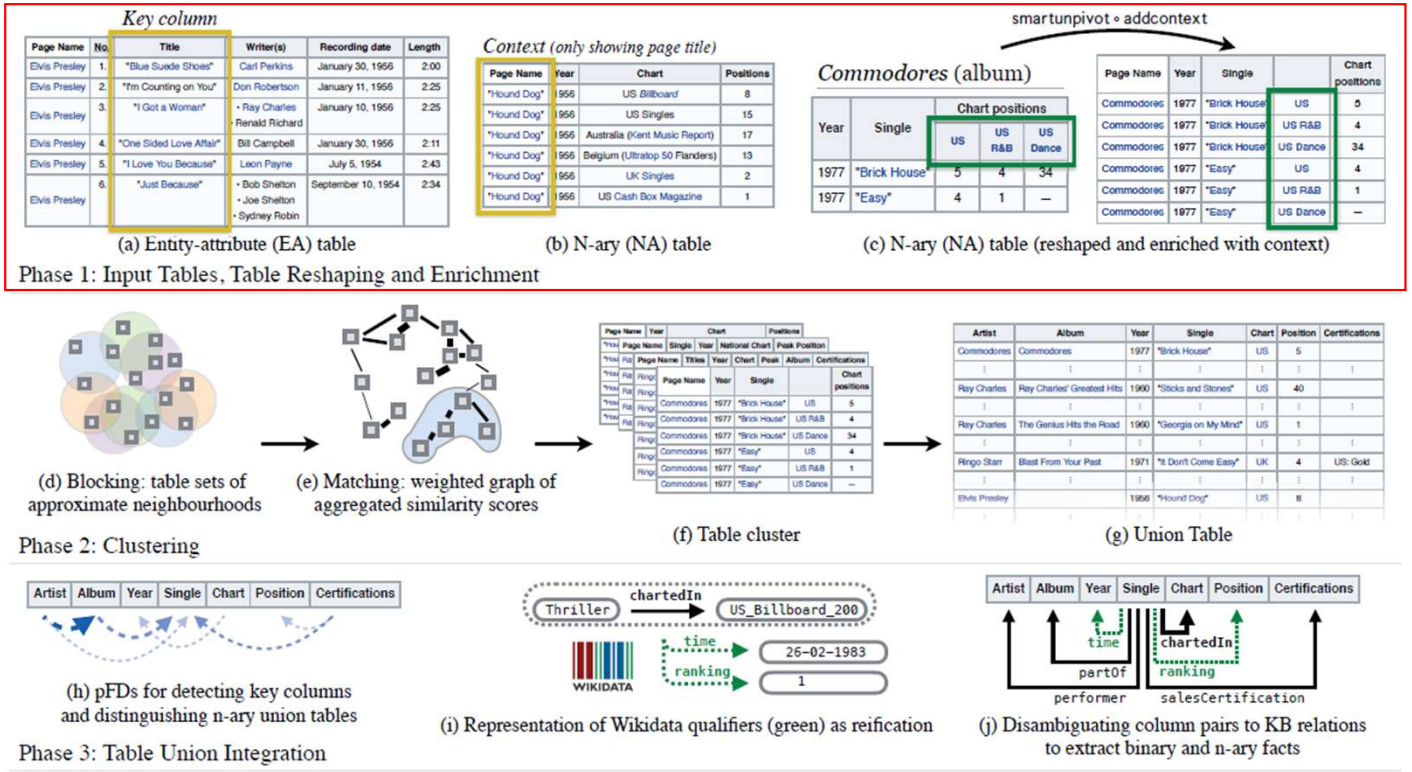


Figure 1: Schematic overview of our pipeline. In Phase 1 (a-c), we process the set of all Wikipedia tables to clean up editorial structures (Section 3.1). In Phase 2 (d-g), we cluster them to form larger union tables (Section 3.2). In Phase 3 (h-j), we integrate them with Wikidata and extract binary and n-ary facts (Section 3.3)

- 위키피디아에서 비슷한 정보를 담고있는 테이블들은 다양한 형태를 하고있음
- 이를 해결하기 위해 테이블의 "정규화"를 진행하고 정규화는 세 단계로 이루어짐 (Merging table chunks, Table Unpivoting, Adding Contextual Information)

2. Our Approach

3.1 Table Reshaping and Enrichment

- Merging table chunks

  - 위키피디아 contributor들이 가시성이나 추가 설명을 위해 테이블에 붙인 추가 셀들은 해석 과정에서 신뢰도를 떨어뜨림
  - Heuristics H_1, H_2, H_3 를 통해 수행함
  - H_1: 짝수번째 row의 cell들(row index i = 2, 4, . . .)이 모든 column을 span하면 해당 cell들은 이전 row의 추가적인 정보를 담은 cell이라고 판단하고, 추가 column을 만든 뒤 i번째 row에 있는 정보를 i-1번째 cell에 옮김
  - H_2: H_1이 적용되지 않고 테이블의 마지막 row에 모든 column을 span하는 cell들이 존재하면 footnote라고 가정하고, 기존 row를 삭제하고 해당 정보를 “footnote”라는 타입의 contextual informatio으로 테이블에 추가함
  - H_2: H_1이 적용되지 않고 모든 column을 span하는 다수의 cell들이 존재하면 해당 cell들은 자기 밑에 있는 cell들에 대한 추가 정보를 제공하는 cell이라고 가정하고, 추가 column을 만든 뒤에 밑에 있던 row부터 추가 정보를 기입하고 앞서 얘기한 cell같은 부분이 나오면 과정을 다시 반복하거나 나오지 않으면 테이블의 마지막까지 추가정보를 기입하게 됨

2. Our Approach

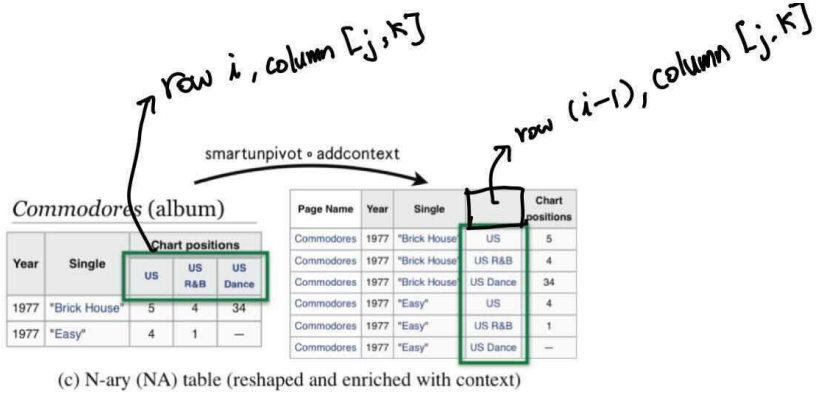
- Table Unpivoting

  - 왼쪽 테이블을 오른쪽 테이블로 바꾸기 위한 unpivoting이 필요함
  - 해당 과정은 두가지 문제가 있음
    - 1. 어떤 테이블을 input으로 받았을 때 수평으로 근접한 header cell을 구분하는 procedure를 정의하는 것
    - 2. 그림에 나와있는 US, US R&B, US Dance와 같은 정보들을 기입할 new column header를 추출하는 것

  - 첫번째 문제는 6개의 Boolean function을 통해 해결함
    - 각 function은 테이블을 input으로 받고(input table T), encoded heuristic이 input의 cell과 매치가 되면 True를 반환함
    - 각 row별로 모든 Boolean function과 매치해보면서 오른쪽 그림의 예와 같이 unpivoting을 수행하게 됨

$U_1, \ldots, U_6 \in \mathcal{U}$

* Set of Boolean function



(c) N-ary (NA) table (reshaped and enriched with context)

2. Our Approach

- 6개의 Boolean function은 다음과 같음
  - U_1 (nPrefix): cell이 숫자로 시작되면 true 반환
  - U_2 (nSuffix): cell이 숫자로 끝나면 true 반환
  - U_3 (linkAgent): Wikidata의 Agent타입(people & organizations)을 가지고 있는 위키피디아의 hyperlink가 포함된 경우 true 반환
  - U_4 (sRepeated): column의 어떤 interval만큼 span하는 cell이 존재하고 다른 row에 동일한 interval에 똑같은 값이 존재하는 경우 true 반환
  - U_5 (headerLike): body 또는 다른 cell에 의해 span된 header cell에 셀이 테이블 T에 더 자주 나타나는 경우 true 반환
  - U_6 (rareOutlier): T에 있는 테이블의 header에 있는 셀의 빈도가 header의 셀의 평균 빈도보다 작은 표준 편차 둘 이상일 경우 true 반환

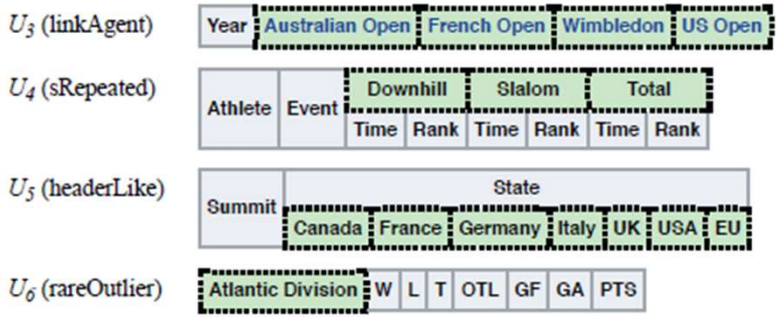


Figure 2: Examples of candidate tables headers for unpivoting. Cells in green are returned by the named heuristic.

2. Our Approach

- Adding Contextual Information

  - Context는 비슷한 entity를 구별하는데 도움을 줄 수 있음
  - 각 테이블에 세가지의 context를 추가함
    - Page title, section title, table caption

  - Procedure addcontext*(T)*
    - Context(T)에 속하는 <X,Y>쌍을 테이블 T에 추가하는 작업
    - 테이블에 header가 X인 extra column을 만들고 cell의 값으로 Y에 해당하는 값을 추가함

Context (only showing page title)

Page Name	Year	Chart	Positions
"Hound Dog"	1956	US *Billboard*	8
"Hound Dog"	1956	US Singles	15
"Hound Dog"	1956	Australia (Kent Music Report)	17
"Hound Dog"	1956	Belgium (Ultratop 50 Flanders)	13
"Hound Dog"	1956	UK Singles	2
"Hound Dog"	1956	US Cash Box Magazine	1

(b) N-ary (NA) table

* addcontext*(T)*의 예

2. Our Approach

3.2 Clustering

- Figure 1의 Table (b)에는 “Chart”라는 열이 존재하지만 Wikidata에는 관련된 부분이 없음 → 동일한 latent relation을 나타내는 테이블 클러스터를 찾아야 함

- 계산의 복잡성때문에 두 단계의 클러스터링 과정을 거침 (d, e)
  - Blocking: k-NN으로 대충 비슷해 보이는 테이블 그룹을 찾음
  - Matching: blocking 과정을 거친 테이블 쌍들의 유사도 점수를 구함

- 최종적으로 union table을 만듦

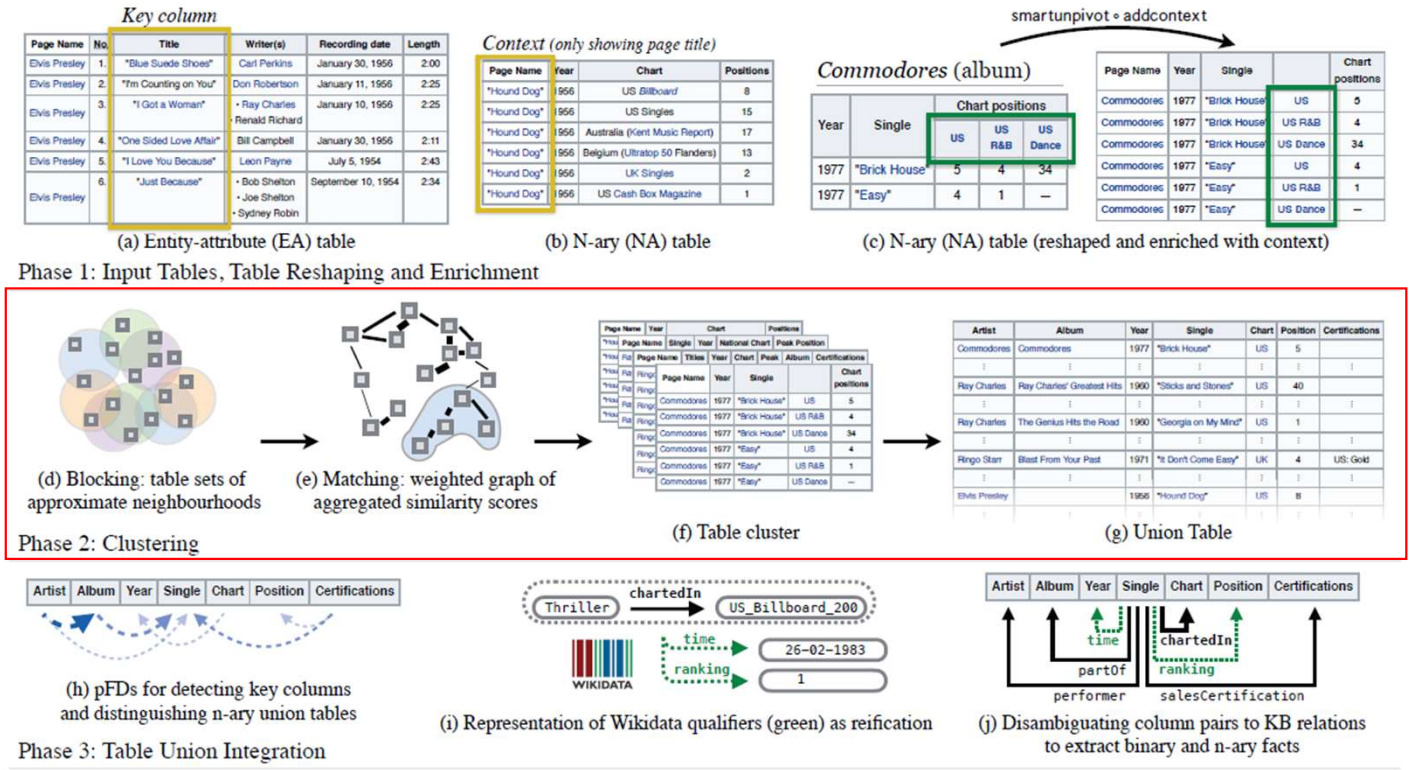


Figure 1: Schematic overview of our pipeline. In Phase 1 (a-c), we process the set of all Wikipedia tables to clean up editorial structures (Section 3.1). In Phase 2 (d-g), we cluster them to form larger union tables (Section 3.2). In Phase 3 (h-j), we integrate them with Wikidata and extract binary and n-ary facts (Section 3.3)

2. Our Approach

- Blocking
  - 유사도를 대략적으로 측정해줄 4개의 index를 사용함 (I_1, I_2, I_3, I_4)
  - 4개의 index를 통해 상위 k개의 table들을 구하고, 구해진 k+1개의 table들은 *table block*이라고 지칭함 (k=100)
  - I_1, I_2는 Locality Sensitive Hashing(LSH)과 MinHash를 같이 사용함
    - Table cell값이 많이 겹침 → 두 table이 비슷함
    - I_1은 table header, I_2는 table body를 고려하여 LSH index를 구함
    - MinHash: 두 테이블의 cell값들의 자카드 유사도를 구함
  - I_3, I_4는 table의 header row와 body column을 임베딩해 k-NN을 통한 유사도를 측정함
    - Word embedding은 pretrained GloVe 사용
    - I_3은 header vector, I_4는 column vector로써 indexing함
  - 계산된 index들을 통해 table t와 유사한 ($t \in T$) 상위 k개의 table들을 구함
    - $4(k+1) * |T|$ blocks를 얻음

2. Our Approach

- Matching
  - Matching function f는 cell set 두개를 input으로 받아 서로 겹치는 cell값, 워드 임베딩에 따른 유사도, semantic type을 비교함
    - f_j: Set Similarity → 자카드 유사도를 이용 $f_j(a,b) = \frac{|a \cap b|}{|a \cup b|}$
    - f_e: Word Embedding Similarity → 각 셀의 word embedding값을 바탕으로 positive cosine distance를 이용해 유사도를 구함

      $$f_e(a,b) = \max(0, \frac{\bar{w}(a) \cdot \bar{w}(b)}{\|\bar{w}(a)\|\|\bar{w}(b)\|})$$
    - f_d : Datatype Similarity → cell에서 추출된 pattern을 바탕으로 hyperlink를 이용하게 됨, 이 유사도는 pattern overlap vector의 코사인 유사도를 통해 구해짐 $\mathcal{G} = \langle \mathcal{T}, \mathcal{W} \rangle$

      $$f_c(a,b) = \frac{O(a) \cdot O(b)}{\|O(a)\|\|O(b)\|}$$

- Clustering
  - Blocking과 matching을 통해 구해진 weighted graph G를 Louvain Community Detection을 이용해 통합 가능한 그래프들끼리 묶음

    graph $\mathcal{G} = \langle \mathcal{T}, \mathcal{W} \rangle$

    $\mathcal{T}$ as vertices,

    matrix $\mathcal{W} \in \mathbb{R}^{\mathcal{T} \times \mathcal{T}}$
  - 유사한 테이블끼리의 cluster를 구한 뒤, cluster 안에 속하는 테이블들의 모든 column을 align함
  - 같이 묶인 column을 바탕으로 union table을 만들고, header는 column cluster에서 자주 쓰이는 header cell을 사용함

2. Our Approach

- KB Integration
  - Union table의 타입을 결정하고 정보를 추출함
- Detecting n-ary union tables
  - pFDs를 이용해 EA, NA union table을 구별함
- Entity Disambiguation
  - T의 셀을 T의 유형에 관계없이 K의 entity로 구분함
- Fact Extraction
  - T가 EA table일 경우, pFDs 중 확률이 높은 것을 골라 K에 있는 relation 중 A와 B의 관계를 잘 표현하는 걸 고름
  - T가 NA table일 경우, multiset $\{y : y, c \in N\ (A, B, X, r)\}$에서 가장 자주 나오는 relation r_X를 구함

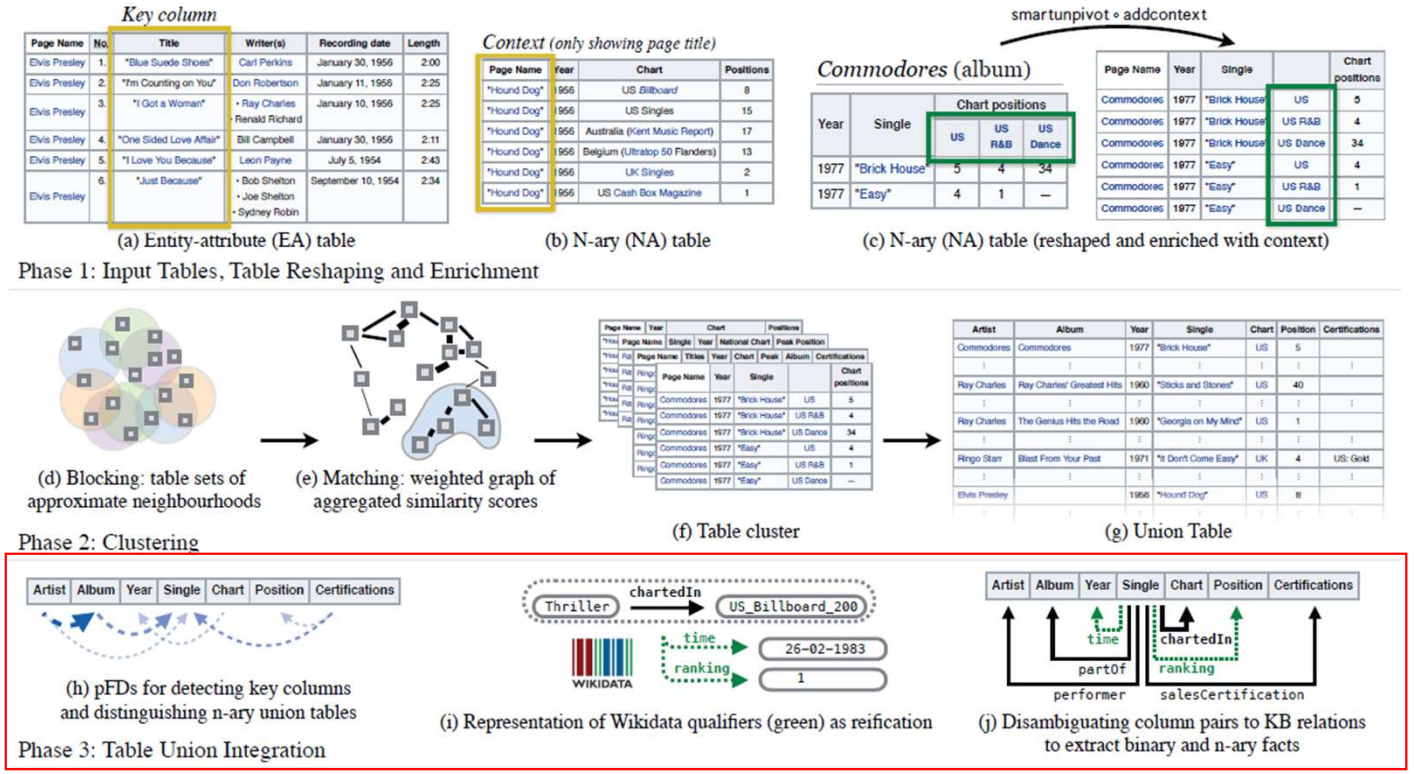


Figure 1: Schematic overview of our pipeline. In Phase 1 (a-c), we process the set of all Wikipedia tables to clean up editorial structures (Section 3.1). In Phase 2 (d-g), we cluster them to form larger union tables (Section 3.2). In Phase 3 (h-j), we integrate them with Wikidata and extract binary and n-ary facts (Section 3.3)

3. Evaluation

- 위키피디아에서 1,535,332개의 table corpus 사용
  - 1,426,303개의 unique tables
  - 330,221개의 unique headers (247,403개, 75%는 한번만 사용된 header)
    - 즉, 1,287,929개의 테이블은 서로 공유하는 header를 가지고 있음
  - 테이블들의 평균 row 개수는 11
  - 2019. 12월부터 생성된 Wikidata를 사용

- 제안된 method에 대한 널리 쓰이는 테스트 방식이 딱히 없기 때문에 human annotator 사용
- 위키피디아 페이지에서 가져온 1,000 random tables (3,449 columns)
- Annotator들이 unpivot해야 할 column을 지정 → “contain names of a related set of concepts that do not describe the content of the column below them.”
- Annotator들의 검증을 거쳐 최종적으로 151개의 테이블이 나옴
- 151개의 테이블을 클러스터링 한 후, union table이 나오게 됨
- Annotator들은 union table 중 query table과 동일한 relation을 표현할 수 있는 union table을 고르게 됨 (기준: 해당 테이블의 모든 행이 쿼리 테이블에 추가되도 이상하지 않은지)

3. Evaluation

Heuristic	Pr.	Re.	F_1
headerLike	0.30	0.33	0.31
numPrefix	0.73	0.27	0.39
sRepeated	**0.89**	0.12	0.21
numSuffix	0.52	0.09	0.16
linkAgent	0.70	0.05	0.09
rareOutlier	0.14	0.02	0.04
All Heuristics	0.44	**0.88**	**0.58**

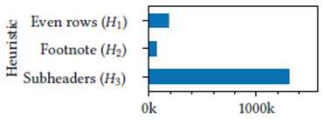
(a) Precision, Recall and F_1 of unpivot detection heuristics

Reshape	Clusters	ARI	AMI	H.	C.	V.
-	One per table	0.00	0.00	1.00	0.85	0.92
-	Same Header	0.55	0.55	1.00	0.90	0.95
Model	Same Header	0.55	0.56	1.00	0.90	0.95
Oracle	Same Header	0.56	0.57	1.00	0.90	0.95
Model	Only Header Sim.	0.83	0.77	0.97	0.95	0.96
Model	Only Body Sim.	0.69	0.67	0.99	0.93	0.96
Model	Both Sim.	**0.88**	**0.85**	0.99	0.97	0.98
Model	Oracle	0.93	0.94	1.00	0.98	0.99

(b) Clustering ablation for different configurations

	Accuracy			
Method	EA	NA	Both	EA pct.
Non-num. baseline	0.29	0.10	0.39	90%
Entity baseline	0.24	0.20	0.43	74%
Ours	**0.45**	**0.77**	**0.66**	36%

(c) Key column prediction accuracy for EA tables, NA table detection accuracy, both tasks combined, and the EA table percentage predicted (EA pct.). The true percentage of EA tables in our annotated sample is 35%



(d) # cells processed by heuristics in mergechunks

	Tables	Columns
Individual	173 (17%)	267 (8%)
Union	193 (33%)	419 (17%)

(e) Number of matched tables and columns in annotated sample

		Existing Facts in $\mathcal{K}$		New Facts	
	Precision	#	Context	#	Context
Binary	0.65	4,708,865	33%	15,800,524	29%
N-ary	**0.92**	2,122,518	22%	6,901,790	19%
All	0.74	**6,831,383**	30%	**22,702,314**	26%

(f) Relation matching precision, number of matched and new facts, and percentage of matched and new facts that involve the table context

Figure 3: Experimental results. The best outcomes (except the annotations-based oracle) are reported in boldface

- Figure 3d: mergechunks 작업을 통해 1,554,692개의 cell을 바로잡아서 후속 작업을 원활하게 함
- Figure 3a: 933,949개의 각기 다른 테이블에 속한 260,528 unique header가 unpivot 처리됐고, heuristic 당 점수도 계산되었음.
- Figure 3b: 제안된 모델과 oracle을 이용한 다양한 클러스터링 방식의 조합을 비교해놓음 → Header similarity("Only Head Sim.")만 이용해 클러스터링한게 body similarity("Only Body Sim.")만 이용한것보다 성능이 좋았음 → 둘다 고려한 클러스터링이 gold standard로 사용된 human annotation에 따른 oracle 모델에 가장 가까웠음
- Figure 3c: key-column과 n-ary table detection의 성능을 측정하고 두개의 baseline과 비교함 → "Non-numeric" baseline은 최소 95%의 고유 값을 포함하는 가장 오른쪽 non-numeric 열을 선택하고, "Entity" basesline은 같은 방식으로 entity를 포함하는 열을 선택하는데 조건에 맞는 열이 없으면 해당 테이블을 n-ary 테이블로 분류함 → EA, NA 두 경우 모두 논문에서 제안한 방식이 우수한 성능을 보여줌
- Figure 3e: individual table보다 union table에서 relation identification을 하는 것이 fact extraction 하기에 낫다는 것을 증명함
- Figure 3f: 논문에서 제안된 방식이 binary fact보다 n-ary fact를 발굴하는데 더 적합했고, "Context"의 비율이 크면 클수록 좋음

4. Conclusion

- 장점
  - 더 많은 테이블을 다룰 수 있는 unpivoting 기반의 새로운 heuristics를 제안함
  - 클러스터링 과정의 Louvain algorithm을 제외하고 모든 단계는 scalable함

- 한계
  - Non-Wikipedia table에 대한 실험은 수행되지 않음