PPDS - Duplication detection

Noah Kogge, Berra Akbas, Jan Dangberg | Praktikum | 08.07.2024

Agenda

1. Introduction

1.1 Definition

Duplication Detection

„Detection method for detecting ***similar*** *or* ***one to one*** *entries in a Database“*

Fact:
- 10 % of customer records are duplicates

Origin:
- **Entry errors:** Typographical / Inconsistent formatting / Humans errors
- **Data integration:** Merging / Integrating / Unification

Problems:
- **Data quality issues:** Accuracy / Inconsistency
- **Perfomance degradation:** Query performance / System overhead

2. Problem Statement

2.1 Main issues in duplication detection

Customer Data 1

id	f_name	l_name	b_place
1	jan	danberg	münchen
2	noah	kogge	berlin

Customer Data 2

id	f_name	l_name	b_place
1	jann	dangberg	munich
2	berra	akbas	-

Merged Data

id	f_name	l_name	b_place
1	jan	danberg	münchen
2	noah	kogge	berlin
3	jann	dangberg	munich
4	berra	akbas	-

Duplicate(fuzzy)
or too large $\rightarrow$ n^2

2. Problem Statement

2.2 Task Overview

Z1.csv (1,223,167 tuples / 28,077 matching pairs)

id	title	-	-	-	-

⟺

ZY1.csv

lid	rid

Z2.csv (1,215,130 tuples / 961,970 matching pairs)

id	name	price	brand	description	category

⟺

ZY2.csv

lid	rid

Infrastructure:

- def **generating_blocking_key**(row: Series) → **distinctive**, **clean**
- def **create_blocks**(input_df: DataFrame)
- def **create_matches**(blocks: dict, input_df: DataFrame)

3. Methodologies

3.1 Data preprocessing

id	title
1	„Lenovo - buy laptop 4GB 30Ram perfect business cf2242342 | eBay“
…	-
32	Amazon - 14´ 13““ - | good
45	„Lenovo - laptop 4GB 30Ram cf2wz2467we | hot.com“
58	„Lenovo camera laptop 2000GB 3Ram cf2wz2467we | rebuy.com“
…	-
70	Amazon.com - | (ver y nice) | - / Lenovo Thinkpad 350Gb 12Ram

„Identify **important / dominant** patterns (e.g. sellers, brand, device etc)“

3. Methodologies

3.1 Data preprocessing

id	title
1	„Dell dell netbook. 12.24 400GB 20Ram tech bui, eBay |- „13@insane cndition“

Text standardization (A-Z → a-z, (-,|,@,“) → (´ ´), good → (´ ´))

↓

Error correction(bui → buy, cndition → condition)

↓

Semantic mapping (insane / perfect / awesome → good)

id	title
1	dell netbook 400gb 20ram ebay

3. Methodologies

3.2 Blocking- and Ascii key generation

Pattern initialization:

```
'brands': re.compile(r'\b(acer|panasonic|toshiba|hp|sony|lenovo')
```

Attribute extraction:

```
def find_brands(text: str) -> str:
    if patterns['brands'].findall(text):
        return " ".join(sorted(set(patterns['brands'].findall(text))))
    else:
        return ''
```

Example:

```
text1 = "lenovo dell tablet good quality c23423c23542"
result1 = find_brands(text1)
print(f"Text 1 Brands: '{result1}'")  # Ausgabe: 'dell lenovo'
```

3. Methodologies

3.2 Blocking- and Ascii key generation

Perform blocking key:

```python
def generate_blocking_key_name(row: pd.Series) -> tuple:
    # Convert the title to a string and lowercase
    title = str(row['title'])
    low_title = title.lower()
```

```python
# Initialize dictionaries to store patterns and corresponding row indices
pattern2id_1 = defaultdict(list)
```

```python
# Generate initial blocking key based on sorted words in the title
pattern2id_1[" ".join(sorted(low_title.split()))].append(row.name)
```

```python
# Extract brands from the cleaned title
brands = find_brands(low_title)
if brands:
    pattern2id_1[brands].append(row.name)
```

```python
return pattern2id_1
```

→ add more blocking keys

3. Methodologies

3.2 Blocking- and Ascii key generation

Ascii key:

Id	f_n	l_n
1	john	smith
2	Jane.*	doe
3	!jane,	doe20
4	jonas	bum

Preprocess

Id	f_n	l_n
1	john	smith
2	jane	doe
3	jane	doe
4	jonas	bum

| j + o + h + n |

[431,549]

[414, 312]

[414, 312]

[539, 324]

Block 1: [431549]

Block 2: [312414]

Measure similarity [2],[3]

Block 3: [324539]

3. Methodologies

3.3 Summary

D1

id	title
1	Dell 12Gb, tablet
2	Acer buy13cam
3	Acer good 13'cam
4	12 Gb on“dell`-tab

Pattern / Attribute extraction

⇩

Ascii / Blocking key

⇩

Create blocks

Block 1: (*acer*)+(*cam*)

id	title
2	Acer buy13cam
3	Acer good 13cam

Block 2: (*dell*)+(*tablet*)

id	title
1	Dell 12Gb tablet
4	12Gb dell tab

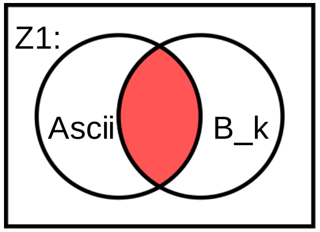


Fig. 1

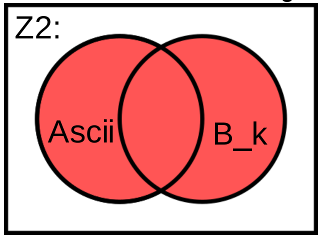


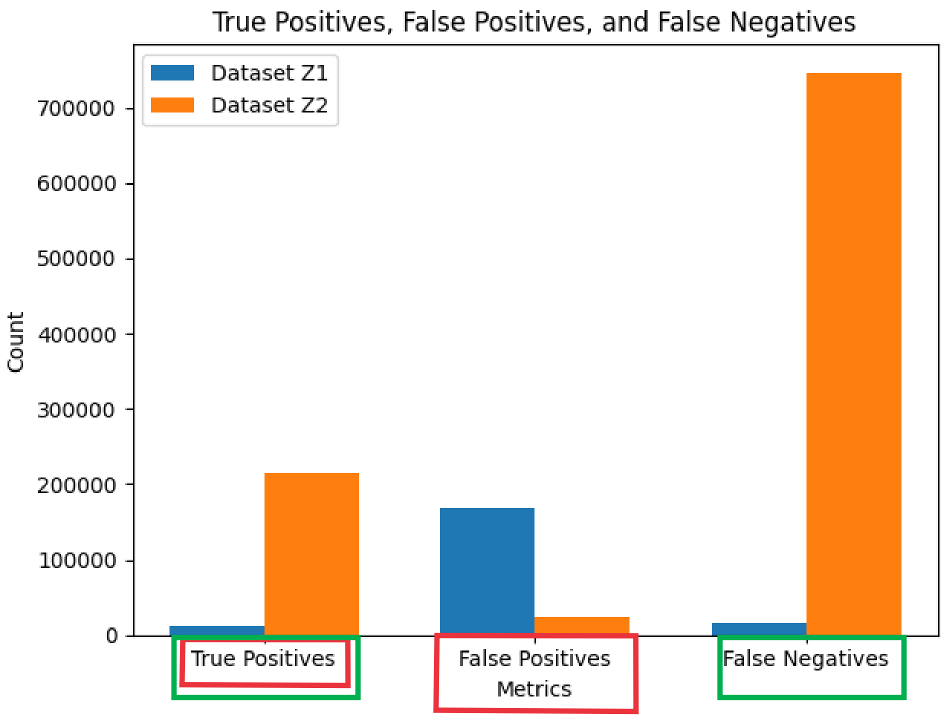
Fig.2

⇩

Match → Evaluate

4. Evaluation

4.1 Results



Environment:
- **Device:** MacBook Pro
- **Chip:** Apple M3 Pro
- **Memory:** 18 GB

Benchmarks:

	Z1	Z2
Jac_sim	0.8	0.75
Precision	0.07	0.90
Recall	0.45	0.22
F1 Score	0.12	0.36
Time total	2214.63 sec	

4. Evaluation

4.1 Results

	Z1	**Z2**
Jac_sim	0.8	0.75
Precision	0.07	0.90
Recall	0.45	0.22
F1 Score	0.12	0.36
Time total	2214.63 sec	

Manually Adjusted (dependent on the dataset)

**Patterns too broad,
capturing all potential matches**

**Patterns too narrow,
covering small subset**

Improvements:

- **Z1**: Precise blocking keys (reduce false positives)
- **Z2**: Broaden blocking patterns (reduce false negatives)

4. Evaluation

4.1 Result

Observation:

```
695913,PANSONIC ASPIRE LENOVO VOLOGY 15.6 DUO EDEN 2GB I7 | 10.1 I7 NVIDIA 2.60 (4
765460,Pansonic Aspire Lenovo Vology 15.6 Duo Eden 2GB i7 | 10.1 I7 NVIDIA 2.60 (4
684204,"Lenovo VivoBook - Mobile 17 14"" Edition"
988519,"LENOVO VIVOBOOK - MOBILE 14"" EDITION"
```

Detected but not **provided** in ground truth

Precision for Z1 **cannot** be increased

4. Evaluation

4.2 Reflection

Takeaway:

- **Balance** between recall, precision and similarity thresholds
- Improve **blocking strategies** (Combine more keys, relaxing key criteria)
- Enhance **scalability** (Progressive classification, different similarity joins or ML)

Our issues:

- **Definition** of **duplicate** (Dataset-dependent, fuzzy duplicates)
- Perfect **preprocessing** (Denoise entries universally, regex extraction, limitations)
- Function **testing** (Inapplicable on whole dataset due to inconsistencies)
- Dominant **pattern filtering** (Extraction of most frequent occurring attributes)

5. References

Fig 1,2

- https://de.wikipedia.org/wiki/Mengenlehre (last visited: 25.06.2024)

- https://dbgroup.ing.unimore.it/sigmod22contest/leaders.shtml (last visited: 01.07.2024)

- https://mboehm7.github.io/teaching/ss24_ppds/05_Experiments.pdf (last visited: 02.07.2024)

- https://bigdama.github.io/teaching/teaching_materials/01_Introduction_D2IP.pdf (last visited: 04.07.2024)

- https://bigdama.github.io/teaching/teaching_materials/02_Duplicate_Detection.pdf (last visited: 03.07.2024)

- https://academic.oup.com/database/article/doi/10.1093/database/baw164/2870676 (last visited: 20.06.2024)

- https://link.springer.com/chapter/10.1007/978-3-642-22913-8_18 (last visited 29.06.2024)

- https://systematicreviewsjournal.biomedcentral.com/articles/10.1186/2046-4053-4-6 (last visited 03.07.2024)