PPRL:
PRIMAT Toolbox

Di Mario Cristiano

Università di Modena e Reggio Emilia

Sistemi Informativi e Web Semantico

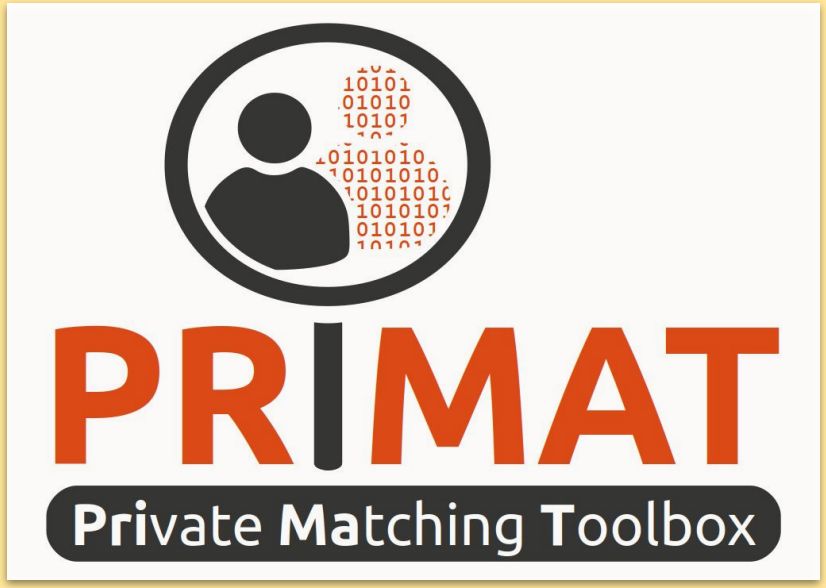
Introduction to PPRL

In many cases, data owners are allowed to provide their data for data integration only if there is sufficient protection of sensitive information to ensure the privacy of individuals (such as patients of a hospital or clients of a facility).

For example, in medical research, data from different sources (e.g., data from different hospitals) must be matched to study possible correlations between diseases without revealing the identity of individual patients.

Privacy Preserving Record Linkage (PPRL) addresses this problem by providing techniques for matching different records while preserving their privacy and allowing data from different sources to be combined to improve data analysis and research.

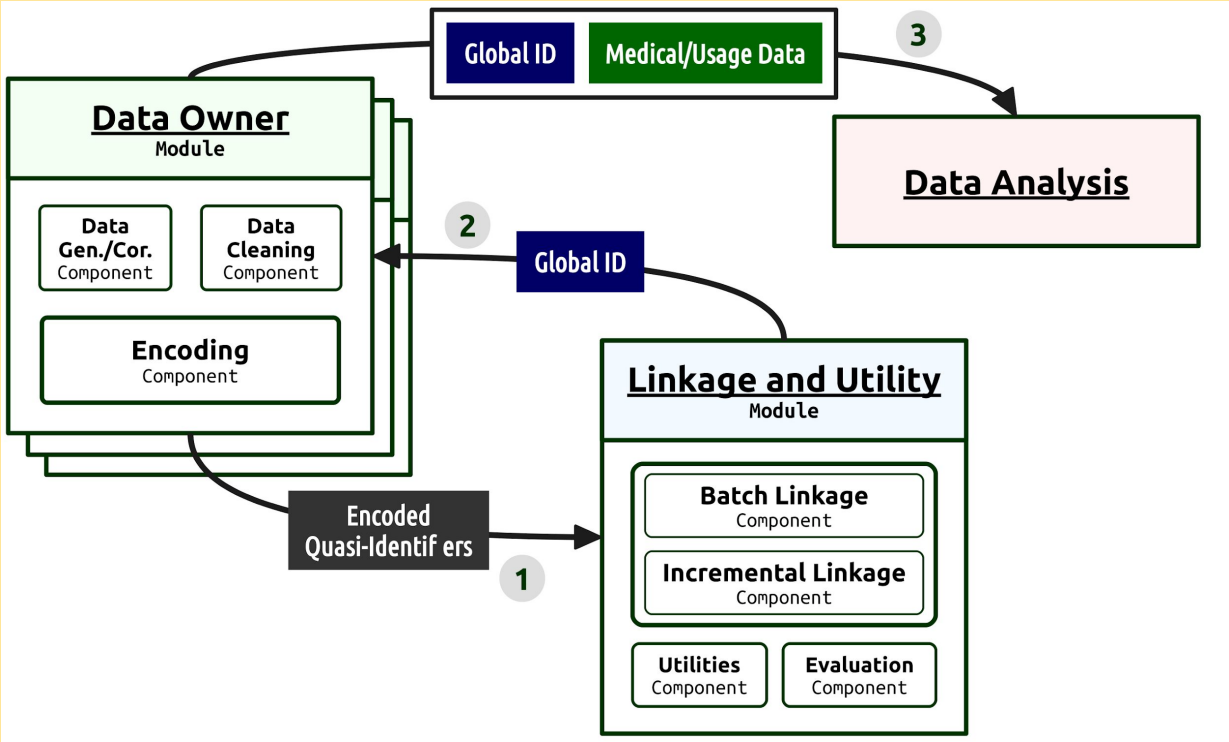
PRIMAT



PRIMAT is an open source toolbox for the definition and execution of PPRL workflows. It offers several components for data owners and the central linkage unit that provide state-of-the-art PPRL methods, including Bloom-filter-based encoding and hardening techniques, LSH-based blocking, metric space filtering, post-processing and more.

PRIMAT is developed by the Database Group of the University of Leipzig, Germany.

PRIMAT Description



Test Example

Input Dataset A: 5000 record Dataset B: 5000 record

Esempio
rec_id, given_name, surname, street_number, address_1, address_2, suburb, postcode, state, date_of_birth, soc_sec_id

rec-1070-org, michaela, neumann, 8, stanley street, miami, winston hills, 4223, nsw, 19151111,5304218
rec-1016-org, courtney, painter, 12, pinkerton circuit, bega flats, richlands, 4560, vic, 19161214,4066625

Data Cleaning Accent Remover Special Character Remover Lower Case Normalizer Umlaut Normalizer
Trim Normalizer

Test
Funzione di Similarità: Jaccard Similarity
Threshold: 0.8
True Positive (TP): 4950
False Positive (FP): 50
Precision: 0.9990
Recall: 1
F-measure: 0.9994
Match completati correttamente: 4950 / 5000

Custom PPRL

LOAD DATASETS: Esempio 3

```
[72] 1 datasetName = "dataset_febrl3.csv"
     2
     3 Table = pd.read_csv(datasetName, encoding = 'unicode_escape').astype('string')
     4 Table['id']=Table['rec_id']
```

```
[90] 1 Table
```

	rec_id	given_name	surname	address_1	address_2	suburb	postcode	state	date_of_birth	soc_sec_id	id
0	rec-1496-org	mitchell	green	wallaby place	delmar	cleveland	2119	sa	19560409.0	1804974	rec-1496-org
1	rec-552-dup-3	harley	mccarthy	pridhamstreet	milton	marsden	3165	nsw	19080419.0	6089216	rec-552-dup-3
2	rec-988-dup-1	madeline	mason	hoseason street	lakefront retrmnt vlge	granville	4881	nsw	19081128.0	2185997	rec-988-dup-1
3	rec-1716-dup-1	isabelle	<NA>	gundulu place	currin ga	utakarra	2193	wa	19921119.0	4314184	rec-1716-dup-1
4	rec-1213-org	taylor	hathaway	yuranigh court	brentwood vlge	<NA>	4220	nsw	19991207.0	9144092	rec-1213-org

```
1 Evaluation(GoldStandard, SM_J_sim_Evaluation)
```

	MT	TP	FP	FN	P	R	F
0	813	787	26	843	0.968	0.4828	0.6443

```
1 ### ENCODE TABLE A
2
3 DA.drop_duplicates(DA.columns[1:3], keep='first', inplace=True) #Index(['given_name', 'surname'], dtype='object')
4
5 for col in DA.columns[1:10]:
6   DA = DA[DA[col].notna()]
7
8 mixColumnsA = DA.columns[1:10] #Index(['given_name', 'surname', 'address_1', 'address_2', 'suburb', 'postcode','state', 'date_of_birth', 'soc_sec_id'],dtype='object')
9 DA['mix'] = ' '
10
11 for x in list(mixColumnsA) :
12   DA['mix'] += DA[x] + ' '
13
14 DA['bf'] = DA.apply(lambda row: ApplyBloomFilter(row), axis=1)
15
16 DA_ENC = DA[[DA.columns[len(DA.columns)-3],DA.columns[len(DA.columns)-1]]]  # DA encoded : ('id', 'bf')
17
18
19 ### ENCODE TABLE B
20
21 DB.drop_duplicates(DB.columns[1:3], keep='first', inplace=True) #Index(['given_name', 'surname'], dtype='object')
22
23 for col in DB.columns[1:10]:
24   DB = DB[DB[col].notna()]
25
26 mixColumnsB = DB.columns[1:10] #Index(['given_name', 'surname', 'address_1', 'address_2', 'suburb', 'postcode','state', 'date_of_birth', 'soc_sec_id'],dtype='object')
27 DB['mix'] = ' '
28
29 for x in list(mixColumnsB) :
30   DB['mix'] += DB[x] + ' '
31
32 DB['bf'] = DB.apply(lambda row: ApplyBloomFilter(row), axis=1)
33
34 DB_ENC = DB[[DB.columns[len(DB.columns)-3],DB.columns[len(DB.columns)-1]]]  # DB encoded : ('id', 'bf')
```

References

- Scientific paper: Privacy Preserving Record Linkage (Rainer Schnell)

- Scientific paper: PRIMAT: A Toolbox for Fast Privacy-preserving Matching (Martin Franke, Ziad Sehili, Erhard Rahm)

- PRIMAT: https://git.informatik.uni-leipzig.de/dbs/pprl/primat

- PRIMAT Application: https://github.com/gen-too/primat

- https://www.boozallen.com/insights/ai/privacy-preserving-record-linkage.html

- https://www.sciencedirect.com/science/article/abs/pii/S0306437921001526

- https://github.com/data61/anonlink-entity-service

- https://github.com/DuncanSmith147/pseudonymization

- My Code + Presentation: https://github.com/mariocris/SIWS.git