

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di ingegneria Enzo Ferrari

Corso di laurea Ingegneria Informatica

Documentazione e aggiunte alla libreria per l'entity resolution SparkER

Relatori:

Prof.ssa Sonia Bergamaschi

PhD Luca Gagliardelli

Candidato:

Arturo Bianchi

Matricola n° 132615

ANNO ACCADEMICO 2020/2021



SOMMARIO

- Entity resolution applicato a una situazione reale
- SparkER, cos'è e quali limitazioni presenta
- L'aggiunta di nuovi formati
- Semplificazione del workflow
- Documentazione dei metodi

ENTITY RESOLUTION

- L'entity resolution (ER) è un insieme di tecniche per identificare tutte le istanze di un oggetto, indipendentemente dal formato o dalla fonte.
- Presenta il problema della complessità quadratica.
- Non scala bene nell'ambito dei big data.

SCENARIO DI UTILIZZO REALE

- Creare un record unico in cui i profili corrispondenti allo stesso oggetto abbiano uno stesso identificatore;
- Utilizzare uno strumento di deduplication o linkage per sfruttare le descrizioni delle entità.

Catalog A

PID	Title	Description
P123X	Syringe 10x10	Syringe 10 ml 10 pack
P123Y	Syringe 10x100	Syringe 10 ml 100 pack
P456A	Insulin needle 4x10	Hypodermic insulin needle 4 mm 10 pack
...

Catalog B

ID	Name	Description
1	Syringes 10 ml	Syringe 10 ml x 10 pieces
2	Syringes 10 ml big pack	Syringe 10 ml x 100 pieces
3	Small needle	Needle for insulin 4 mm 10 pieces
...

SPARKER

- Una libreria open-source per python sviluppata in UNIMORE dal DBGROUP, già presente su GitHub
- Utilizza come ambiente il framework di Apache Spark, che consente di lavorare in modo efficiente nell'ambito dei big data, permettendo la scalabilità dell'ER
- SparkER implementa le tecniche di entity resolution, non presenti di default su Spark.

COSA SI PUÒ MIGLIORARE

- Numero limitato di formati per caricare i profili;
- Workflow molto lungo con metodi che richiedono molti parametri;
- Mancanza di documentazione dei metodi;

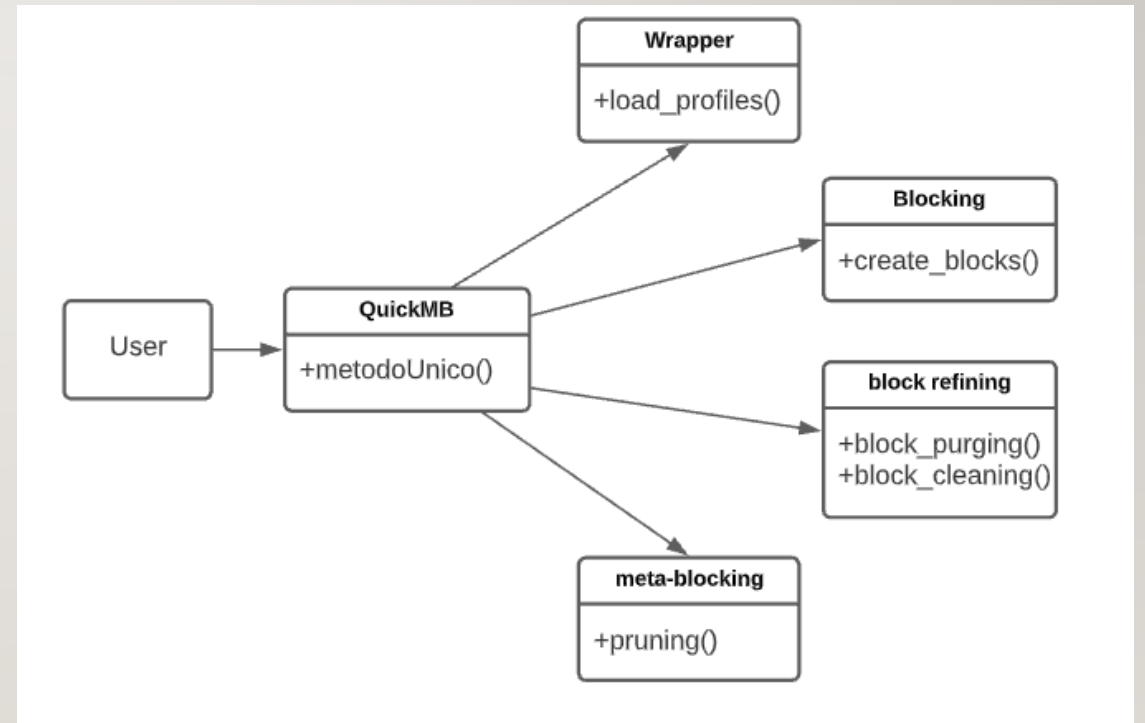
AGGIUNTA DI NUOVI FORMATI

- Utilizzo del framework di pandas;
- Utilizzo del wrapper per i dataframe di pandas;
- Creazione di dataframe da dataset xlsx(excel) utilizzando l'engine openpyxl;
- Ottenimento di dataframe da query sql usando l'engine sqlalchemy.



FACADE PATTERN

- Utilizzo del design pattern facade (facciata) per implementare un'interfaccia più semplice;
- Implementazione per data linkage (due dataset), data deduplication (dataset singolo) e BLAST.



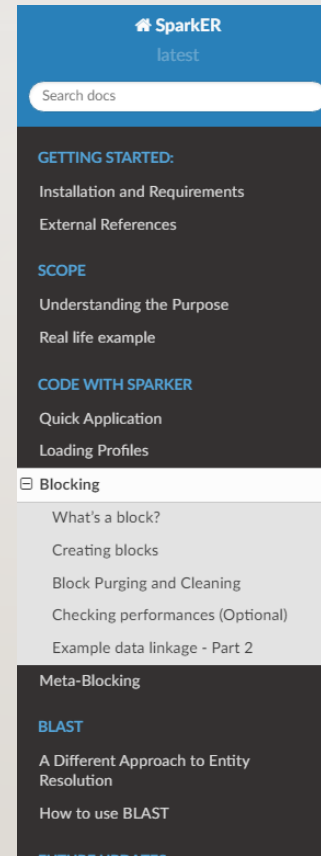
STRUMENTI PER LA DOCUMENTAZIONE

- Servizio di hosting con Read The Docs
- Sphinx e reStructuredText come strumento e linguaggio di markup per la realizzazione delle pagine.



STRUTTURA DELLA DOCUMENTAZIONE

- La documentazione spiega i metodi per l'implementazione del blocking/meta-blocking/BLAST (in inglese).
- Documentazione più pratica che teorica.
- Suddivisa in 5 sezioni, partendo dall'installazione della libreria e arrivando all'analisi dei risultati dei processi.



The screenshot shows the navigation menu of the SparkER documentation website. At the top, there is a blue header with the SparkER logo and the word 'latest'. Below the header is a search bar labeled 'Search docs'. The main menu is divided into several sections: 'GETTING STARTED' (Installation and Requirements, External References), 'SCOPE' (Understanding the Purpose, Real life example), 'CODE WITH SPARKER' (Quick Application, Loading Profiles), 'Blocking' (What's a block?, Creating blocks, Block Purging and Cleaning, Checking performances (Optional), Example data linkage - Part 2), 'Meta-Blocking', and 'BLAST' (A Different Approach to Entity Resolution, How to use BLAST).

and you can understand that, if we are working with peta/exabytes of data, ER doesn't scale very well. But if we can only reduce the comparisons for the number of profiles in every block, then the comparisons number is much much lower, and since comparing can't take forever since "time is money", blocking is a really useful tool.

Creating blocks

SparkER offers different techniques for blocks creation, we're going to show them all, so you can use which ever you prefer; they may differ in performance, but that's up to the dataset composition or other factors.

```
blocks = sparker.Blocking.create_blocks(profiles, separator_ids=None, keys_to_exclude=None,
                                     attributes_to_exclude=None,
                                     blocking_method=sparker.BlockingKeysStrategies.token_blocking
                                     )
```

The function above takes as input the set of profiles obtained as a result of the loading profiles step, the separator ids (you may or may not have to input it, depends if you are doing deduplication or linkage), some optional parameters to filter keys or attributes you want to bypass and returns a set of blocks, containing profiles that share the same keys. This is a standard token blocking strategy, is general purpose and the easiest and quickest to use, but the `blocking_method` parameter is the one that you can change to use a different blocking strategy.

Other strategies are:

- `BlockingKeysStrategies.token_blocking_w_attr`
- `BlockingKeysStrategies.ngrams_blocking`

While both strategies share the same parameters as the above method, n-gram blocking actually require an additional parameter `ngram_size`, which gives the size of the n-grams used for the strategy, by default it's set to 3.

CONCLUSIONI

Lavorare con SparkER ha permesso:

- Un primo approccio al mondo dei big data, in particolare alla teoria dietro l'entity resolution;
- Studio e utilizzo di framework per l'analisi dei dati come Spark e librerie software come pandas;
- Scoperta e studio di alcuni degli strumenti nell'ambito dell'open-source.