SCHEMA MATCHING

Progetto Finale Big Data - AA20/21

Gruppo:
The Hadoopers

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Emanuele Spezia
Giuseppe Brescia

OBIETTIVI

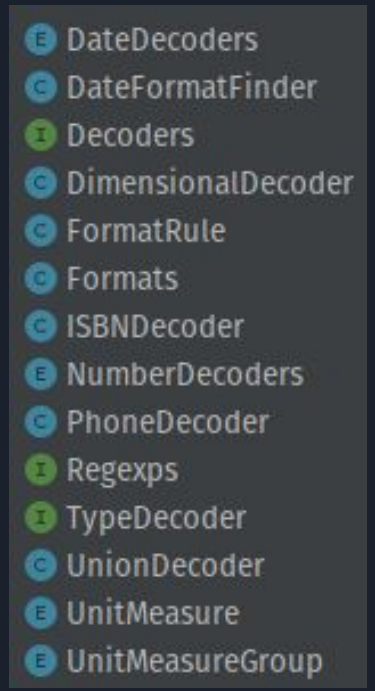
1. Effettuare queries al DBMS NoSQL Clickhouse
2. Memorizzazione ed estrazione dati in forma vettoriale
3. Ricerca relazioni fra vettori di dati estratti da siti web utilizzando un algoritmo di matching
4. Test risultato finale

CLASSI DI SUPPORTO

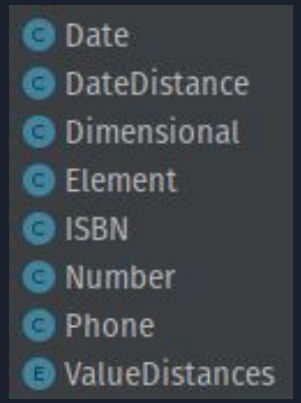
DECODERS	VALUE	VALUE DISTANCES
• DateDecoder • Dimensional Decoder • ISBNDecoder • NumberDecod • PhoneDecoder	• Date • Dimensional • ISBN • Number • Phone	• *DATE_DISTANCE* • *DIMENSIONAL_DISTANCE* • *ISBN_DISTANCE* • *NUMBER_DISTANCE* • *HAMMING*

CLASSI DI SUPPORTO

DECODERS



VALUE



VALUE DISTANCES

QUERIES E MEMORIZZAZIONE DATI

Classe: RetrievingData.java

```
@Controller
public class RetrievingData {

@RequestMapping(value = "/retrieveData", method = RequestMethod.GET)

public Vector<Object> retrieveData(@RequestParam("id") String id,
@RequestParam("page_url") String page_url)
```

QUERIES E MEMORIZZAZIONE DATI

Classe: RetrievingData.java

```
final String URL = "jdbc:clickhouse://localhost:8123/extracted_data";
final String USER = "default";
final String PASSWORD = "";

Vector<Object> vec = new Vector<>();
Connection connection = null;
Statement statement = null;
```

QUERIES E MEMORIZZAZIONE DATI

Classe: RetrievingData.java

```
// register JDBC drive
Class.forName("ru.yandex.clickhouse.ClickHouseDriver" );
// Open the connection
connection = DriverManager. getConnection(URL, USER, PASSWORD);
System.out.println("Connected database successfully" );
// Execute the queries
statement = connection.createStatement() ;
System.out.println("Execute the query" );
String sql = "SELECT * FROM extracted_data." + id + " WHERE page_url = '" +
page_url + "'";
System.out.println(sql);
ResultSet rs = statement.executeQuery(sql) ;
ResultSetMetaData rsmd = rs.getMetaData() ;

while (rs.next()) {
   for (int i = 1; i < rsmd.getColumnCount() ; i++) {
       System. out.println(i);
       vec.add(rs.getObject(i)) ;
   }
}
```

ALGORITMO DI MATCHING

Classe: MatchingFinder.java

```
@Controller
public class MatchingFinder {

@RequestMapping(value = "/", method = RequestMethod.GET)
public String formThreshold() { return "formT"; }

@RequestMapping(value = "/application", method = RequestMethod.GET)
public String findMatch(@RequestParam("threshold") String threshold,
Vector<Object> input_1, Vector<Object> input_2) {
```

ALGORITMO DI MATCHING

Classe: MatchingFinder.java

```
Map<Integer, List<Element>> mapResult = new HashMap<>();
int t = 0;
Double threshold_final = Double.parseDouble(threshold);
TypeDecoder[] decoders = {Decoders.ISBN_DECODER, Decoders.NUMBER_DECODER,
       Decoders.PHONE_DECODER, Decoders.DATE_DECODER};
UnionDecoder ud = new UnionDecoder(decoders);
```

ALGORITMO DI MATCHING

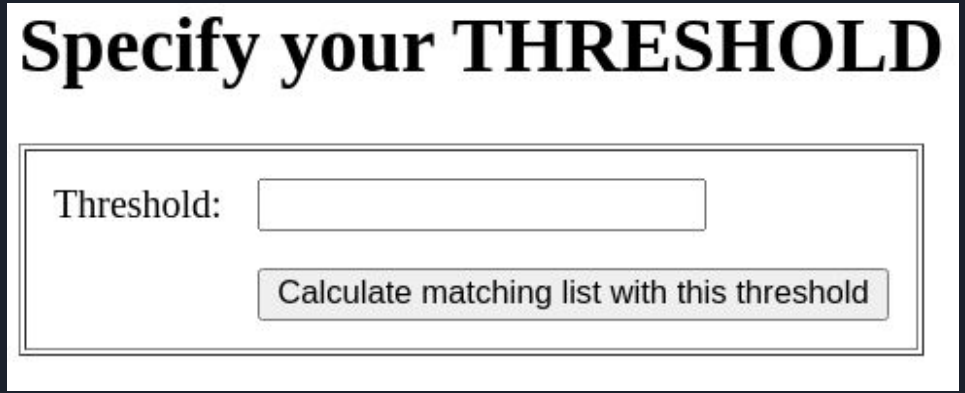
Classe: MatchingFinder.java

```
for (int i = 0; i < input_1.size(); i++) {
   for (int j = 0; j < input_2.size(); j++) {

       /* ISBN */
       if (ud.decode(input_1.get(i).toString()) instanceof ISBN &&
               ud.decode(input_2.get(j).toString()) instanceof ISBN) {
           if ((ValueDistances.ISBN_DISTANCE.distance(input_1.get(i).toString(),
input_2.get(j).toString())) <= threshold_final) {
               List<Element> matchingList = new ArrayList<>();
               matchingList.add(new Element("ISBN", input_1.get(i).toString()));
               matchingList.add(new Element("ISBN", input_2.get(j).toString()));
               mapResult.put(t, matchingList);
               t = t + 1;
           }
       }
```

TEST E RISULTATI

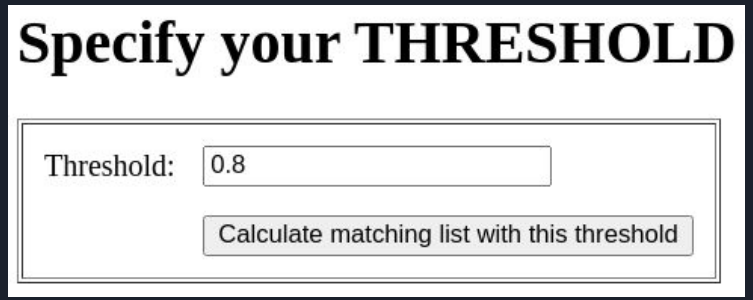
Servizio REST accessibile da localhost:8080/
Rimanda al servizio REST localhost:8080/application quando si inserisce la soglia prescelta



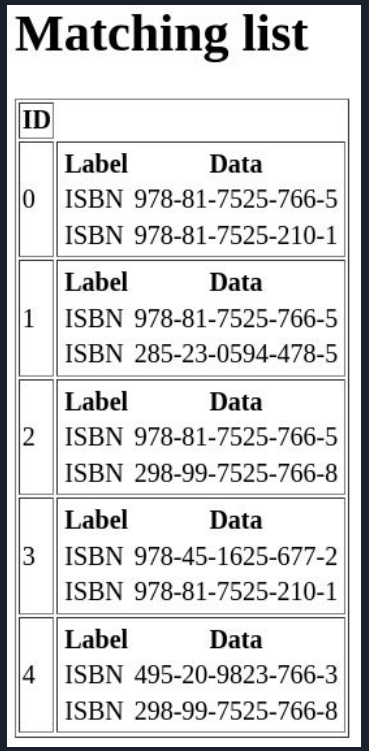
Servizio: /

TEST E RISULTATI

Servizio: /



Servizio: /application

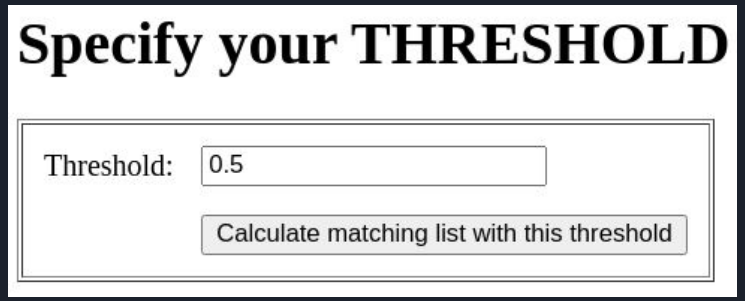


Matching list

ID	
0	**Label** **Data** ISBN 978-81-7525-766-5 ISBN 978-81-7525-210-1
1	**Label** **Data** ISBN 978-81-7525-766-5 ISBN 285-23-0594-478-5
2	**Label** **Data** ISBN 978-81-7525-766-5 ISBN 298-99-7525-766-8
3	**Label** **Data** ISBN 978-45-1625-677-2 ISBN 978-81-7525-210-1
4	**Label** **Data** ISBN 495-20-9823-766-3 ISBN 298-99-7525-766-8

TEST E RISULTATI

Servizio: /



Servizio: /application

VERSIONE FINALE

UTENTE INSERISCE (MICROSERVIZIO RICHIEDE):

- SOGLIA
- ID (identificatore tabella ClickHouse)
- PAGE URL (url della pagina appartenente alle linkage page)

RITORNA:

- LISTA DI MATCHING (lista con etichette e dati di coppie di elementi con valore di matching settato per etichetta)

GRAZIE PER L'ATTENZIONE

Progetto su GitHub:
https://github.com/EmanueleSpezia/SchemaMatching

ROMA TRE
UNIVERSITÀ DEGLI STUDI

EMANUELE SPEZIA
GIUSEPPE BRESCIA