Building a Query Engine for a Corpus of Open Data

Mauro Pelucchi, Giuseppe Psaila and Maurizio Toccu
University of Bergamo - DIGIP, Viale Marconi 5, I-24044 Dalmine (BG), Italy

Keywords: Retrieval of Open Data, Blind Querying, Single Item Extraction.

Abstract: Public Administrations openly publish many data sets concerning citizens and territories in order to increase the amount of information made available for people, firms and public administrators. As an effect, Open Data corpora has become so huge that it is impossible to deal with them by hand; as a consequence, it is necessary to use tools that include innovative techniques able to query them.
In this paper, we present a technique to select open data sets containing specific pieces of information, and retrieve them in a corpus published by a portal of open data. In particular, users can formulate structured queries blindly submitted to our search engine prototype (i.e., being unaware of the actual structure of data sets). Our approach reinterpret and mixes several known information retrieval approaches, giving at the same time a database view of the problem. We implemented this technique within a prototype, that we tested on a corpus containing more that over 2000 data sets. We noted that our technique provides focused results w.r.t. the baseline experiments performed with Apache Solr.

1 INTRODUCTION

Public Administrations are publishing more and more data sets concerning their activities and citizens. Since these data sets are open, i.e. publicly available for any interested people and organizations, they are called *Open Data*. The reader should not confuse the concept of Open Data for Public Administration (which is the context of this paper), with the concept of *Linked Open Data*, i.e., the ability of describing the content of web sites linked each other through RDF (Resource Description Framework).

Public Administrations extended there web sites with *Open Data Portals*; typically (see (Höchtl and Lampoltshammer, 2016)) they exploit two frameworks named *Socrata* and *CKAN*, that can be viewed as CMS (Content Management Systems) to support the publication of data sets within an Open Data Portal.

Data sets published by Open Data Portals are *Structured* data sets: in the simplest form, they are CSV files with a fixed number of fields, with field names reported in the first line. Alternatively, they are often represented as vectors of JSON objects, without nesting (they must be represented in a flat way), and (less frequently) as structured XML documents (without exploiting nesting capabilities of XML). In practice, data sets are always thought as tables: an item in the data set (CSV row, JSON object, XML element) represents a table row.

Nowadays, the number of data sets published by a single *Open Data Portal* is very large and is continuously increasing. Often, a corpus published by one single publishing system can contain several thousands of data sets.

People (typically analysts) that need to find the right data sets among the mass of available data sets in an Open Data Portal have to work hard to find what they look for. They need a powerful search tool to query the above corpus.

We observed that users, a priori, do not know the actual structure of thousands of heterogeneous data sets collected in the corpus. So, they can only blindly query the corpus, making some hypothesis about property or field names. Furthermore, we also observed that often they are not even interested in an entire data set, but only in some rows (items) among those contained in the data set, i.e., only those satisfying a given selection condition. For example, a user/analyst might want to get information about high schools located in a given city named “My City”, being interested in their *name*, *address* and *reputation*. The query mechanism should be able to focus the search on those data sets that actually contain the items of interest, and retrieve them. However, at the same time, it should be able to *extend the search* to

Pelucchi, M., Psaila, G. and Toccu, M.
Building a Query Engine for a Corpus of Open Data.
DOI: 10.5220/0006308801260136
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 126-136
ISBN: 978-989-758-246-2

those data sets that could potentially contain items of interest, by considering similar names in the corpus catalog (i.e., the list of data sets in the corpus and their fields). For example, a data set with field "city_name" or simply "city" in place of "cityname".

Therefore, in this paper, we propose a technique for querying a corpus of Open Data Sets on the basis of queries that users blindly formulate on the corpus. The technique mixes a query expansion mechanism, which relies on string similarity matching, and a keyword selection mechanism based on the concepts of informativeness and representativeness.

The paper is organized as follows. Section 2 gives some preliminary definitions. Section 3 formally defines the problem we are dealing with. In Section 4, the overall query process is presented. Then, Section 5 presents the query assessment step and the keyword selection step. Section 6 introduces the query expansion step. Finally, Section 7 presents the steps concerning open data set retrieval and filtering. The experimental evaluation is shown in Section 8, while Section 9 discusses some related work. Section 10 draws the conclusions.

2 PRELIMINARY DEFINITIONS

In order to facilitate the paper comprehension, we give some preliminary definitions.

Definition 1: **Data Set.** An *Open Data Set ods* is described by a tuple

ods: <ods_id, dataset_name, schema, metadata>

where *ods_id* is a unique identifier, *dataset_name* is the name of the data set (not unique); *schema* is a set of fields names, *metadata* is a set of pairs *(label, value)*, which are additional meta-data associated with the data set. □

Definition 2: **Instance and Items.** With *Instance*, we denote the actual content of data sets. It is a set of *items*, where an item is a row of a CSV file or data table, as well as a flat object in a JSON vector. □

Many open data portals provide open data sets in several formats. Commonly, they adopt CSV (comma separated values), but it is becoming frequent the adoption of JSON (JavaScript Object Notation); XML is also used, but less frequently.

The adoption of JSON is motivated by the fact that, now, it has become the de facto standard for data sharing in the world of Web Services. In fact, it is easy to read, much lighter than XML and JavaScript (the programming language used for developing client-side web applications) natively interprets JSON data.

However, in the open data world, it is used only as an alternative representation for flat tabular data: a CSV file can always be transformed into a vector of JSON objects, where all objects in the vector has the same structure, i.e., the same fields and fields have only simple values (there are not nested objects).

Consequently, JSON capabilities to describe nested complex structures, as well as collections of heterogeneous objects, are not exploited in this context.

To illustrate, consider Figure 1. We report a fragment of the open data set named *Local Authority Maintained Trees* published on the `London Data Store`, that is the official open data portal of London City. Notice how a flat CSV file (in the upper part of the figure) can be easily transformed into a vector of JSON objects (lower part of the figure). As the reader can see, the price to pay is that the JSON version is more verbose, since field names mast be repeated for each object. However, in the optic of integrating the items extracted from several open data sets, having different structures, into a unique collection of results, JSON is the natural representation format for our tool.

Definition 3: **Global Meta-data.** The list of fields of items in the open data set *ods* is *ods.schema*. We call the triple *<data_set_name, schema, metadata>* the *Global Meta-Data* of the data set. □

Definition 4: **Corpus and Catalog.** With $C = \{ods_1, ods_2, \ldots\}$ we denote the corpus of *Open Data sets*. The *catalog* of the corpus is the list of descriptions, i.e., global meta-data, of open data sets in *C* (see Definition 1). □

Definition 5: **Query.** Given a *Data Set Name dn*, a set *P* of field names (properties) of interest $P = \{ pn_1, pn_2, \ldots\}$ and a selection condition *sc* on field values, a query *q* is a triple *q: <dn, P, sc>*. □

Definition 6: **Query Term.** With *Query Term* (*term* for simplicity) we denote a data set name *q.dn*, a field name appearing in *q.P*, in *q.sc*, a constant appearing in *q.sc*. □

3 PROBLEM DEFINITION

Consider an open data corpus published by an *Open Data Portal*, i.e., a system that publishes open data sets.

Queries allow users to discover, in the corpus *C*, those data sets, possibly having the specified name *q.dn* or a similar one, that contain items (e.g., rows or JSON objects) with the desired field names *q.P* (or similar) and can be actually filtered on the basis of

CSV Version

```
ID,Borough,Species name,Common name,Display name,Ox,Oy
1,Barking ,ACER PSEUDOPLATANUS ’BRILLIANTISSIMUM’,,Maple,548320.8,189593.7
2,Barking ,TAXUS BACCATA FASTIGIATA,,Other,548297.9,189590.2
```

JSON Version

```
[
 {
   "ID": 1,
   "Borough": "Barking ",
   "Species name": "ACER PSEUDOPLATANUS ’BRILLIANTISSIMUM’",
   "Common name": "",
   "Display name": "Maple",
   "Ox": 548320.8,
   "Oy": 189593.7
 },
 {
   "ID": 2,
   "Borough": "Barking ",
   "Species name": "TAXUS BACCATA FASTIGIATA",
   "Common name": "",
   "Display name": "Other",
   "Ox": 548297.9,
   "Oy": 189590.2
 }
]
```

Figure 1: Fragment from the data set *Local Authority Maintained Trees* published in the London Data Store.

the selection condition *q.sc*, so that users can obtain filtered items. By relying on the definition of a *relevance measure* for data sets (in Section 7), we can define the overall problem.

Problem 1: Given a *Relevance Measure* $rm(ods) \in [0,1]$ of a data set $ods \in C$, and a minimum threshold $th \in [0,1]$, a data set *ods* is *relevant* if $rm(ods) > 0$ and $rm(ods) \geq th$.
The result set $RS = \{o_1, o_2, \dots\}$ of query q is the set of items (rows or JSON objects) o_i such that ods_j is a relevant data set, $o_i \in Instance(ods_j)$ and o_i satisfies the selection condition *q.sc*. □

Example 1: The sample query about schools in Section 1 becomes:

q :<dn=`Schools`,
    P= {`Name, Address, Reputation`},
    sc= (`City="My City" AND`
          `Type="High School"`) >. □

4 PROCESSING STEPS

The contribution of this paper is a technique to solve Problem 1.

The proposed technique is built around a query mechanism based on the Vector Space Model (Manning et al., 2008), encompassed in a multi-step process devised to deal with the high heterogeneity of open data set and the blindly query approach (the user does not know the actual schema of data sets in the corpus). Recall that, in our vision, the user performs a *blindly query*, but asks for specific field names and for items (rows or JSON objects) that satisfy a precise selection condition. The system is fully aware of data set schemata, but due to their heterogeneity it is not possible to rely only on exact matching. Thus, it is necessary to focus the search, identifying those terms in the query that mostly represent the query essence and the capability of the query to select items within data set instances. At the same time, it is necessary to expand the search space, considering terms which are similar to those in the query, in order to address the fact that users are unaware of actual schemata of data sets.

Moving from the above considerations, we defined the *steps* performed by our technique.

- Step 1: *Query Assessment.* Terms in the query are searched within the catalog. If some term t is missing, t is replaced with term $\bar{t}$, the term in the meta-data catalog with the highest similarity score w.r.t. t. We obtain the *assessed query* $\overline{q}$.

- Step 2: *Keyword Selection.* Keywords are selected from terms in the assessed query $\overline{q}$, in order to find the most representative/informative terms for finding potentially relevant data sets.
- Step 3: *Neighbour Queries.* For each selected keyword, a pool of similar terms which are present in the meta-data catalog are identified, in order to derive, from the assessed query $\overline{q}$, the *Neighbour Queries*, i.e., queries which are similar to $\overline{q}$. Both $\overline{q}$ and the derived neighbour queries are in the set Q of queries to process. For a neighbour query, the set of relevant keywords is obtained from the set of relevant keywords for $\overline{q}$ by replacing original terms with similar terms.
- Step 4: *VSM Data Set Retrieval.* For each query in Q, the selected keywords are used to retrieve data sets based on the Vector Space Model (Manning et al., 2008): in this way, the set of possibly relevant data sets is obtained.
- Step 5: *Schema Fitting.* The full set of field names in each query in Q is compared with the schema of each selected data set, in order to compute the relevance measure *rm*. Data sets whose schema better fits the query will be more relevant than other data sets.
- Step 6: *Instance Filtering.* Instances of relevant data sets are processed in order to filter out and keep only the items (rows or JSON objects) that satisfy the selection condition.

5 QUERY ASSESSMENT AND KEYWORD SELECTION

In Step 1, query q is assessed against the meta-data catalog, in order to check for the presence of terms in q: if all terms in q are present in the catalog, the assessed query $\overline{q}$ is the same query q; otherwise, the assessed query $\overline{q}$ is derived from q by substituting the missing terms with the most similar and representative terms in the catalog. Since the concept of representativeness is introduced hereafter in this section, we will formally define the notion of most similar and representative term in Section 5.2.

5.1 Keyword Selection

Consider the assessed query $\overline{q}$. Terms appearing in $\overline{q}$ are not equally important/informative for the query. If the set of terms used to retrieve possibly relevant data sets contains terms that are not important or much less important than others, in Step 4 we can retrieve data sets that may be not actually relevant for the query. Thus, we adopted a *keyword selection technique* that extracts, from the assessed query $\overline{q}$, the set of (meaningful) keywords *K(*$\overline{q}$*)*.

We were inspired by the technique introduced in (Liu et al., 2006) for the context of classical search engines. However, we introduced several variations (structure of the graph and values of scores) to cope with peculiarities of our problem.

The keyword selection technique proceeds as follows:

1. The *Term Graph* is built, where each node corresponds to a term in the assessed query $\overline{q}$ and an edge connects two terms and weights their relationship (see Figure 3 for a sample graph discussed after the keyword selection algorithm reported in Figure 2).
2. An iterative algorithm:
   (a) Chooses the most relevant term.
   (b) Modifies the weights of not selected terms, propagating a decreasing reduction of informativeness to neighbours of the selected term.
3. Only most informative terms are kept and inserted into the keyword set *K(*$\overline{q}$ *)*.

Term Graph Construction. Each node in the Term Graph corresponds to a term in the query. There are three types of nodes: one *Data Set Name Node*, corresponding to the data set name $\overline{q}$*.dn*; *Field Nodes*, corresponding to field names in $\overline{q}$*.P* or in $\overline{q}$*.sc*; *Value Nodes*, corresponding to constants appearing in comparison predicates in $\overline{q}$*.sc*.

Nodes are connected each other by an edge if terms are related to each other. In particular: a field node is connected with the data set name node; two field nodes are connected if they are the operands of the same comparison predicate in $\overline{q}$*.sc*; a field node and a value node are connected if the field (described by the field node) is compared with the constant value (described by the value ode) by a comparison predicate in $\overline{q}$*.sc*.

Each node (term) has two scores: a *representativeness* score, denoted as *r_score*, and an *informativeness* score, denoted as *i_score*.

Definition 7: **Representativeness Score.** The representativeness score *r_score* denotes the capability of the term to find a relevant data set, i.e., how much it represents a specific subset of mapped data sets. In particular, we consider two components for representativeness: *Intrinsic Representativeness* and *Extrinsic*

Representativeness. The former is related to the presence of the term in the query; we denote it as *irep(t)* (where *t* is a term). The latter is related to the presence of the term in global meta-data of data sets, thus it measures the actual capability of the term to focus the search; we denote the extrinsic representativeness as *erep(t)*. Both *irep* and *erep* are defined in the range $[0,1]$.

The *r_score* of a term is obtained by combining the intrinsic and the extrinsic representativeness, by means of the average of them, i.e., *r_score*$=(irep(t)+erep(t))/2$. This way, both contribute and balance each other. □

Intrinsic Representativeness. Depending on the role played by a term in the query, we established different values for the intrinsic representativeness.

- The intrinsic representativeness of the *Data Set Name* is *irep(t)*$=1.0$ if there exists a data set $\overline{q}.dn$ in the corpus *C*; otherwise, *irep(t)*$=0.0$.
- The intrinsic representativeness of a *Field Name* is *irep(t)*$=0.5$ if there exists at least a data set that has a field named *t*; otherwise, *irep(t)*$=0.0$.
- The intrinsic representativeness for string value nodes and date value nodes, is *irep(t)*$=0.5$, while it is *irep(t)*$=0.2$ for numerical value nodes.

Extrinsic Representativeness. The approach we adopted to define the extrinsic representativeness, is inspired by the *idf* (*inverse document frequency*) measure (Manning et al., 2008). Its classical definition is *idf=log(D / d(t))*, where *D* is the total number of documents, while *d(t)* is the number of documents that contain term *t*. The lower the number of documents that contain *t*, the higher the value of *idf*, because term *t* is more capable to focus the search on relevant documents.

Moving from this idea, we defined the extrinsic representativeness in order to be in the range $[0,1]$ and to be suitable for our context.

Definition 8: **Normalized Data Set Frequency.** Consider the corpus *C* of open data sets and its cardinality $|C|$. Consider a term *t*, the set *Matched(t)* of data sets that contain term *t* in global meta-data (such that *Matched(t)*$\subseteq$*C*) and its cardinality $|Matched(t)|$.

The *Normalized Data Set Frequency* of term *t* is defined as

$$ndf(t) = log_2\left(\frac{|Matched(t)|}{|C|}+1\right)$$

that is defined in the range $[0,1]$.

If term *t* is contained in the global meta-data of all data sets, *ndf(t)*$=1$ (the argument of the logarithm is 2. If term *t* is not contained in any global meta-data, *ndf(t)*$=0$, because the argument of the logarithm is 1. □

Definition 9: **Extrinsic Representativeness.** Given a term *t* and its normalized data set frequency *ndf(t)*, its *Extrinsic Representativeness* is defined as:

$$erep(t) = 1 - ndf(t) \quad \text{if } ndf(t) > 0$$
$$erep(t) = 0 \quad \text{if } ndf(t) = 0 \quad \square$$

The rationale is the following: if a term is very frequent, its extrinsic representativeness is close to 0, because it is not able to focus the search. In contrast, not frequent terms are able to focus the search, and their extrinsic representativeness is close to 1.

Definition 10: **Informativeness Score.** The informativeness score *i_score* measures the degree of information that could be obtained by selecting a term as keyword. At the beginning, an initial value is set: we chose *i_score*$=1.0$ for data set name nodes and field nodes that appear in the selection condition $\overline{q}.sc$. We chose *i_score*$=0.8$ for value nodes and field nodes appearing only in $\overline{q}.P$. In fact, constants are naturally less informative than data set names and field names; furthermore, they are less likely to appear in meta-data of data sets, thus the data set retrieval step based on VSM would result ineffective. The same is for field names not appearing in the selection condition: they are less informative than those appearing in the selection condition.

During the keyword selection process, the selection of a term affects the informativeness of neighbours: in fact, selecting two neighbours may be redundant. Consequently, the procedure will modify the *i_score* values of not selected terms. □

Keyword Selection Algorithm. The algorithm is inspired by the algorithm reported in (Liu et al., 2006), but modified because, in our term graph, edges are neither labeled nor weighted. The main procedure of the algorithm, named *KeywordSel*, is reported in Figure 2.

The algorithm works on the input sets of nodes and edges; notice that nodes have an additional field named *marked*, denoting selected nodes when *true*.

The main cycle from lines 2 to 13, at each step selects the node not yet selected with the highest value for *i_score* + *r_score*, provided that the sum is greater than 1. This is because below this threshold the capability of retrieving significant data sets becomes too low. The *for* loop from line 4 to line 8 looks for the highest *i_score* + *r_score* value (and the corresponding node) in not marked nodes. Then, the *if* instruction at line 9 verifies that the selected node (if any) has the sum *i_score* + *r_score* greater than 1: if so, the node is marked as selected, the corresponding term is inserted into the keyword set *K* and the procedure *UpdateIScore* is called (line 12) to propagate a reduction of informativeness.

```
   Procedure KeywordSel
   Input: $N$ set of nodes, $E$ set of edges
   a node is $n = < t, i\_score, r\_score, marked = false >$
   Output: the set of selected keywords $K$
   begin
1.   $K = \emptyset$; // set of selected keywords
     do
2.     $max\_score$= 0.0; //initial max score
3.     $max\_node$=$null$; //no node selected
4.     for each node $n \in N$ with $n.marked$ =$false$ do
5.       $score$= $n.i\_score + n.i\_score$;
6.       if $score$>$max\_score$ then
7.         $max\_score$=$score$;
8.         $max\_node$= $n$;
         end if
       end for each
9.     if $n \neq null$ and $max\_score > 1.0$ then
10.      $K = K \cup \{n.t\}$; //term selected as keywordd
11.      $n.marked$ =$true$; //node marked as selected
12.      call procedure UpdateIScore($N$, $E$, $n$);
       end if
13.  while $max\_score > 1.0$;
   end procedure
```

Figure 2: Procedure *KeywordSel*, that selects possibly relevant keywords.

Procedure *UpdateIScore* performs the update of the *i_score*, by propagating from the selected node. The propagation mechanism can be described in a recursive way.

- Starting from the selected node s, for each node z' connected with s, the reduction factor is $rf(z') = s.r\_score\ /2/n$, where n is the number of nodes connected with s.
- Consider a node z that just received a reduction factor $rf(z)$ and the set $Out(z)$ of nodes connected to z that have not received yet a reduction factor. For each node $z' \in Out(t)$, the reduction factor is $rf(z') = rf(z)\ /\ 2\ /\ |Out(z)|$. The propagation continues until all nodes have a reduction factor.
- For each node z, the *i_score* is decreased by the reduction factor $rf(z)$, i.e., $z.i\_score = z.i\_score - rf(z)$ (where the lower bound is $z.i\_score = 0$).

This way, the representativeness of the selected node decreases the informativeness of neighbours; the reduction becomes lower and lower as the distance from the selected node increases.

Figure 3.a reports the initial term graph for the (assessed) query in Example 1. Figure 3.b shows how *i_score* values change, showing the propagation from the selected node (*City*) to the other nodes.

5.2 Deriving the Assessed Query

Having defined the concept of *extrinsic representativeness* for a term *t* (denoted as *erep(t)*), we can now describe how query q is assessed into query $\overline{q}$.

- The assessed query $\overline{q}$ is derived from query q by replacing each term *t* non appearing in the meta-data of data sets with a substituting term *t'*.
- First of all, we consider the notion of *string similarity metric* between two terms *t* and *t'*, denoted as *sim(t, t')*$\in [0,1]$. We adopt the well known *Jaro-Winkler* metric (see (Winkler, 1999), (Jaro, 1989)).
- For a pair of terms *t* and *t'*, we define the *suitability* degree as $suitability(t, t') = (sim(t, t') + erep(t'))/2$, i.e., a term *t'* is more suitable to replace the missing term *t* if *t'* is as similar as possible to *t* and, at the same time, it is as representative as possible of data sets (extrinsic representativeness).
- A missing term *t* is replaced, in the assessed query $\overline{q}$, by the most suitable term *t'* in the catalog, i.e., having the greatest *suitability(t, t')*.

The rationale of this choice is simple: we want to substitute missing terms with those that are more likely than others to find relevant data sets, w.r.t. to the query.

6 DERIVING NEIGHBOUR QUERIES

The assessed query $\overline{q}$ could be not enough to find relevant data sets for the user, because he/she writes the query blindly, w.r.t. data set meta-data. As an example, if the user wants to query a field named *latitude*, he/she is usually interested also in data sets having field *lat* (common short-cut for *latitude*). To make the query process effective, we derive, from the assessed query $\overline{q}$, the *neighbour queries nq*, i.e., queries that are close to $\overline{q}$. Both $\overline{q}$ and nq_i are collected into Q, the set of *queries to process*.

Definition 11: **Pool of Alternative Terms.** Consider a term $t \in K(\overline{q})$. We denote as *Alt(t)* the set of alternative terms for *t*.

A term $t' \in Alt(t)$ if: 1) *t'* is in the meta-data catalog; 2) if its string similarity with *t* is grater than or equal to a minimum similarity threshold *th_sim*, i.e., $sim(t, t') \geq th\_sim$; 3) given *max_alt* the maximum number of alternative terms, *t'* is one of the *max_alt* top-most similar terms to *t*. □

In other words, *Alt(t)* contains the *max_alt* terms which are mostly similar to *t*.

Definition 12: **Potential Neighbour Query.** Consider the assessed query $\overline{q}$, with its keyword vector

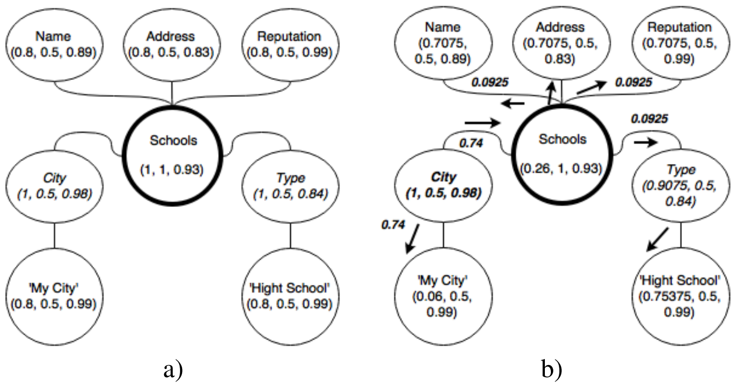

Figure 3: Initial state of the Term Graph (a) and after the first keyword selection (b). The triple in each node is *(i_score, irep, erep)*.

$K(\overline{q})=[t_1, \ldots, t_n]$ and a vector of weights $W(\overline{q})$, where $W(\overline{q})[i]=1$, for each term $t_i=K(\overline{q})[i]$. A potential neighbour query *pnq* and its keyword vector *K(pnq)* are obtained as follows:
- $|K(pnq)| = |K(\overline{q})|$.
- One or more terms t_i in $K(\overline{q})$ are replaced with one term chosen from the set of alternative terms $t^i_j \in Alt(t_i)$. If t_i is not replaced, $K(pnq)[i]=t_i$; otherwise, $K(pnq)[i]=t^i_j$. Correspondingly, the new potential neighbour query *pnq* is obtained by performing the same replacements in the assessed query $\overline{q}$.
- The vector of weights for the potential neighbour query is as follows: if t_i is not replaced, $W(pnq)[i]=1$; otherwise, $K(pnq)[i]=sim(t_i, t^i_j)$. □

Any query obtained by replacement of one or more terms is a potential neighbour query. However, their number could be high and the query could be possibly too far away from the assessed query; therefore, we have to prune those potential neighbour queries that may be too distant, based on the cosine similarity metric applied to the weight vectors.

Definition 13: **Neighbor Queries.** Given the assessed query $\overline{q}$ and a potential neighbour query p, p is considered a *neighbour query* if $CosineSim(W(\overline{q}), W(p)) \geq th\_neighbour$, where $th\_neighbour \in [0,1]$ is a minimum threshold. □

This way, the set Q of queries to process is populated by the assessed query $\overline{q}$ and all neighbour queries.

7 DATA SET RETRIEVAL

Each query $nq \in Q$ has associated the vector *K(nq)* of keywords selected in step 2 (for the assessed query) or in step 3 (for neighbour queries). In step 4, the Vector Space Model (Manning et al., 2008) is applied to retrieve possibly relevant data sets.

The VSM approach relies on the concept of *inverted index* created on the bases of terms in the global meta-data of each data set. Definitely, in our context the inverted index *IN* implements a function $f(t) \rightarrow \{ods\_id_1, ods\_id_2, \ldots\}$, that, given a term t, returns the set of data set identifiers whose global meta-data contain term t.

Consider a vector of keywords $K(nq)=[k_1, \ldots, k_n]$. Then, given a data set identifier *ods_id*, the occurrences of keywords associated with *ods_id* in the inverted index *IN* are represented by the vector $m(ods\_id) = [p_1, \ldots, p_n]$, where $p_j = 1$ if the data set is associated with the keyword, $p_j = 0$ otherwise.

The *Keyword-based Relevance Measure krm(ods, nq)* of a data set *ods* for a query $nq \in Q$ is the average of p_j values. The maximum value of $krm(ods)$ is 1, obtained when the data set is associated with all the keywords, while values less than 1 but greater than 0 denote partial matching.

7.1 Schema Fitting

This step of our technique computes, for each data set retrieved in step 4, the final relevance measure *rm*, by composing the keyword-based relevance measure *krm* of a data set with the fitting degree of its schema w.r.t. the field names in the query. This step is necessary to identify, among all retrieved data sets, those having an instance that can be effectively processed by the selection condition *nq.sc*.

We adopt an approach based on a weighted inclusion measure, where the weight of each field name depends on the role played in the query by the field name.

Definition 14: **Schema Fitting Degree.** Consider a data set *ods* and its schema $ods.schema=<pn_1, pn_2, \ldots>$. Then, consider a query $nq \in Q$ and the set of field names $N = \{n_1, \ldots, n_k\}$ that appear in the selection condition *nq.sc* and/or in the list of field names of interest *nq.P*. For each name $n_i \in N$, a weight $w_i \in [0,1]$ is associated (see Definition 15), depending on the role played by n_i in the query. The *Schema Fitting Degree sfd(ods, nq)* is defined as:

$$sfd(ods, nq) = \sum_{i=1}^{|N|} w_i \times p(n_i) \; / \; \sum_{i=1}^{|N|} w_i$$

where function $p(n_i) = 1$ if there exists a field name $pn_j = n_i$ in *nq.schema*, $p(n_i) = 0$ otherwise. □

The following definition defines how names are weighted.

Definition 15: **Weight.** The weight w_i of a field name $n_i \in N$ is

$$w_i = max(pw_i, scw_i)$$

where pw_i is the weight due to the presence of n_i in the list *nq.P* of fields of interest in the query, while scw_i is the weight due to the presence of n_i in the selection condition *nq.sc*.

- $pw_i = 0.5$ if field name $n_i \in nq.P$ (may be relevant to have this field in the resulting items); $pw_i = 0$ otherwise.

- $scw_i = 1$ if the selection condition *nq.sc* contains one single predicate with n_i, or the predicates in the condition are ANDed each other; $scw_i = 0.7$ if predicates are ORed each other. □

Finally, we define the relevance measure for a data set.

Definition 16: **Data Set Relevance Measure.** Consider a data set *ods* and a query $nq \in Q$. Given its *Keyword-based Relevance Measure krm(ods, nq)* and its *schema Fitting Degree sfd(ods)*, the *Relevance Measure* of data set *ods* w.r.t. query *nq* is

$$rm(ods, nq) = (\alpha - 1) \times krm(ods, nq) + \alpha \times sfd(ods, nq)$$

where $\alpha = 0.6$. The *data set relevance measure rm(ods)* is

$$rm(ods) = max(rm(ods, nq_i)),$$

for each $nq_i \in Q$. □

The rationale is that the schema fitting degree corrects the keyword-based relevance measure. The reason is simple: the keyword-based relevance measure is obtained by querying the inverted index in a blind mode, i.e., irrespective of the fact that a keyword is a data set name, a field name or a word appeared in meta-data. The keyword selection technique partially corrects this blind approach, to discard those data sets that could result accidentally relevant.

Finally, in order to perform item selection from downloaded data sets, how the field names in the query match the actual schema of data sets becomes crucial, because it is impossible to evaluate the selection condition if fields in the condition are not present (all or partially) in the schema of the data sets. This also motivates the choice of $\alpha = 0.6$.

Finally (see Section 3), the minimum threshold *th* is used to discard less relevant data sets and keep only those such that $rm(ods) \geq th$.

8 EXPERIMENTAL EVALUATION

We implemented our technique in a prototype and evaluated the effectiveness of the technique. Here, we report a short evaluation, due to space limitations.

We performed experiments on a set of 2301 open data sets published by the *Africa Open Data* portal[1]. These open data sets are prepared in a non homogeneous way, as far as field names and conventions are concerned. This fact makes the search more difficult than with an homogeneous corpus (such as, e.g., New York City open data).

We performed 6 different queries. They are reported in Figure 4. Hereafter, we describe them.

In query q_1, we look for statistical information about malnutrition in Monbasa or Nairobi or Turkana counties in Kenya. Apart from counties, we are interested in fields that describe underweight, stunting and wasting. In query q_2, we look for information about civil unions since 2012, where the age of both spouses is at least 35. The features of interest are `year`, `month`, `spouse1age` and `spouse2age`. In query q_3, we look for information about availability of water, a serious problem in Africa. In particular, we look for distribution points that are functional (note the condition, where we consider two possible values for field `functional-status` (`functional` and `yes`). In query q_4 we look for information about the number of teachers in public schools, for each county. In query q_5, we look for data about per capita water consumptions at the date of December, 31 2013 (notice that the date is written as `"2013-12-31t00:00:00"`, to stress the similarity search). Finally, query q_6 look for data about orchid export in terms of weight (field `kg`) and `money`.

To perform the evaluation, we downloaded all the data sets and we explored and labelled every data set and item as correct or wrong w.r.t. each query [2].

We performed 9 tests, in order to evaluate the effect of varying some parameters. The maximum number of alternative terms *maxc_alt* was set to 3 for all tests, and the cosine similarity threshold for neighbour queries *th_neighbour* was set to 0.9995 for all

[1] https://africaopendata.org/

[2] We used *Open Refine* (http://openrefine.org) a powerful tool to work with messy data

Table 1: Evaluation of Experimental Results (n.s.i. means *no selected items*).

Recall (%) for Data sets

Query	Test 1	Test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9
q_1	13.33	13.33	13.33	13.33	13.33	13.33	80.00	80.00	80.00
q_2	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_3	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_4	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.67
q_5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_6	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Precision (%) for data sets

Query	Test 1	Test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9
q_1	66.67	66.67	66.67	66.67	66.67	66.67	7.79	8.22	8.22
q_2	50.00	50.00	50.00	1.15	50.00	1.15	1.14	1.15	1.15
q_3	75.00	75.00	75.00	50.00	50.00	37.50	50.00	50.00	37.50
q_4	13.33	13.33	13.33	13.33	11.76	13.33	13.33	11.76	13.33
q_5	100.00	100.00	100.00	100.00	50.00	100.00	6.67	6.25	6.67
q_6	20.00	33.33	33.33	20.00	25.00	25.00	20.00	25.00	25.00

Recall (%) for Items

Query	Test 1	Test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9
q_1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_2	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_3	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00	100.00
q_4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
q_6	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00	100.00

Precision (%) for Items

Query	Test 1	Test 2	test 3	test 4	test 5	test 6	test 7	test 8	test 9
q_1	50.00	50.00	50.00	50.00	50.00	50.00	0.31	0.31	0.31
q_2	89.91	89.91	54.40	89.91	89.91	54.40	89.91	89.91	54.40
q_3	n.s.i.	51.43	51.43	n.s.i.	48.54	48.54	n.s.i.	48.54	48.54
q_4	10.39	14.39	9.72	10.39	9.72	9.72	10.39	9.72	9.72
q_5	100.00	100.00	100.00	100.00	100.00	100.00	24.73	32.85	49.46
q_6	n.s.i.	100.00	50.00	n.s.i.	100.00	50.00	n.s.i.	100.00	50.00

tests. Instead, we evaluated three different values for *th_sim*, i.e., the string similarity threshold for alternative terms, (0.9, 0.8 and 0.7) in combination with three different values for *th*, i.e., the data set relevance threshold (0.3, 0.25, 0.2). The combinations are: for *test1*, *th_sim*= 0.9 and *th*= 0.3; for *test2*, *th_sim*= 0.8 and *th*= 0.3; for *test3*, *th_sim*= 0.7 and *th*= 0.3; for *test4*, *th_sim*= 0.9 and *th*= 0.25; for *test5*, *th_sim*= 0.8 and *th*= 0.25; for *test6*, *th_sim*= 0.7 and *th*= 0.25; for *test7*, *th_sim*= 0.9 and *th*= 0.2; for *test8*, *th_sim*= 0.8 and *th*= 0.2; for *test9*, *th_sim*= 0.7 and *th*= 0.2.

In the first two sections of Table 1, we report recall and precision as far as the retrieval of data sets is concerned, before filtering items. The reader can observe that we often get very high values of recall, even 100%, meaning that we are able to get all meaningful data sets; the data set recall is sensitive to minimum relevance threshold variation only for query q_1, where with *th*= 0.2 the recall significantly improves. Looking at the precision for data sets, results are more various; for query q_2, precision is very sensitive to the variation of thresholds, due to the large amount of false positive downloaded data sets; for query q_6, the variation of minimum thresholds generally improves precision, meaning that we are able to better capture the data sets of interest.

However, the final goal of our technique is to retrieve items that satisfy the selection condition. The third and forth section in Table 1 reports the evaluation of recall and precision for selected items. The reader can notice that in this case things significantly change: recall values are generally very high, meaning that the technique is able to retrieve those (typically few) items of interest, in particular with low settings for minimum thresholds. Looking at precision, we observe that sometimes (query q_4) we get low values (around 10%), meaning that a large num-

Table 2: Comparison of T Test 2 with the base the baseline obtained by means of Apache Solr.

Query	Baseline		Test 2 - Data Sets		Test 2 - Items	
	Recall(%)	**Precision(%)**	**Recall (%)**	**Precision(%)**	**Recall(%)**	**Precision(%)**
q_1	100.00	22.09	13.33	66.67	100.00	50.00
q_2	100.00	1.96	100.00	50.00	100.00	89.91
q_3	100.00	25.00	100.00	75.00	100.00	51.43
q_4	100.00	33.33	66.67	13.33	100.00	14.39
q_5	100.00	100.00	100.00	100.00	100.00	100.00
q_6	100.00	10.00	100.00	33.33	100.00	100.00

```
q1:<dn=nutrition,
    P={county, underweight, stuting,
        wasting},
    sc=(county = "Monbasa"
          OR county = "Nairobi"
          OR county = "Turkana")>
q2 :<dn=civilunions,
    P={year, month, spouse1age,
        spouse2age},
    sc=(year = 2012 AND (spouse1age >= 35
         OR spouse2age >= 35)>
q3 :<dn=wateravailability,
    P={district, location, position,
        wateravailability},
    sc=(functional-status = "functional"
         OR functional-status = "yes" )>
q4 :<dn=teachers,
    P={county, schooltype,
         noofteachers},
    sc=(schooltype="public")>
q5 :<dn=waterconsumption,
    P={city, date, description,
        consumption_per_capita},
    sc=(date = "2013-12-31t00:00:00")>
q6 :<dn=national-export,
    P={commodity, kg, money},
    sc=(commodity = "Orchids")>
```

Figure 4: Queries for the Experimental Evaluation.

ber of false positive items is selected; this is the effect of the Jaro-Winkler string similarity measure adopted to find similar fields.

The goodness of the approach is shown by the experiments with query q_6. The high precision for items obtained in Test 2 (in spite of low precision for data sets) is confirmed looking at downloaded items: many of them have value "ORCHIDS/CYMBIDIUM" for field commodity, instead of value "Orchid" written in the query. Thus, our technique resulted particularly effective in the expansion phase with *th_sim*.
Our technique obtained the best performance for Test 2, with *th_sim=0.8* and t*h=0.3*. The reason is that our approach is greedy during the *VSM Data Set Retrieval*; then, in the *Schema Fitting* step the search space is narrowed and data set instances are downloaded. The filtering (and latter) phase is lazier: so the average precision for items is above of 60%.

To complete the evaluation, we identified a suitable baseline. Among other tools, we decided to adopt *Apache Solr* (Shahi, 2015), which is an open source enterprise search engine, developed on top of *Apache Lucene*.

We indexed the Africa Open Data corpus with Apache Solr and executed the same 6 queries, evaluating both recall and precision for the retrieved data sets (Apache Solr retrieves only data sets, while it is not able to retrieve items).

Table 2 shows the results of our comparison: the first couple of columns reports recall and precision for data sets retrieved by Apache Solr, the second couple of columns reports recall and precision for data sets retrieved by our prototype in Test 2; the third couple of columns reports recall and precision for items retrieved by our prototype in Test 2.

It is possible to see that recall is always 100% for Apache Solr, while in two cases it is not 100% for retrieved data sets in Test 2. In contrast, our prototype always overcomes Apache Solr as far as precision is concerned: this is mainly due to the generation of neighbour queries and keyword selection.

Finally, when we consider retrieval of items (not supported by Apache Solr), out prototype generally behaves very well, with 100% of recall and generlaly very high values for precision (except for query q_4).

9 RELATED WORKS

The research area of this paper is certainly young, and, at the best of our knowledge, we are in the pioneering phase. However, many researchers are reasoning about Open Data management.

In (Khosro et al., 2014) Khosro et al., present the state of art in Linked Open Data (LOD), with issues and challenges. Moreover, the authors motivate this topic by exploiting the projects analysed in the five major computer science areas (Intelligence, Multimedia, Sensors, File System and Library), and present the future trends and directions in LOD.

The Open Data world is related to the Linked Data world. In fact, standardized proposals are typically used to describe published Linked Open Data. The RDF (*Data Set Description Framework*) (Miller, 1998) is widely used for this purpose. Furthermore, the standard query language for RDF is called *SPARQL Standard Protocol and RDF Query Language* (Clark et al., 2008). W.r.t. our proposal, it is very general and devoted to retrieve those data sets with certain features. However, highly skilled people in computer science are able to use it. In contrast, our query technique is very easy for not skilled people and closer to the concept of query in information retrieval.

Similar considerations are done in (Auer et al., 2007), where the authors presents *DBPedia*: the RDF approach is not suitable for non expert users that need a flexible and simple query language.

In this paper we do not consider *Linked Open Dara*: we query a corpus of Open Data Sets, in general not related to each other, thus, not linked at all.

The idea of extending our approach to a pool of federated Open Data Corpora is exciting. A pioneer work on this topic is (Schwarte et al., 2011), but they still rely on SPARQL as query language.

Finally, the heterogeneity of Open Data asks for the capability of NoSQL databases. In (Kononenko et al., 2014), the authors report their experience with *Elasticsearch* (distributed full-text search engine), highlighting strengths and weaknesses.

10 CONCLUSION

This paper presents a technique to retrieve items (rows in CSV files or objects in JSON vectors) contained in open data data sets from those published by an open data portal. User blindly query the published corpus. The technique both focuses the search on relevant terms and expands the search by generating *neighbour queries*, by means of a string matching degree. The experimental evaluation shows that the technique is promising and effective.

New and more extensive experiments will be performed in the future, as well as the technique will be refined and further improved, by adding semantic information to drive the choice of similar terms. In particular, as far as this point is concerned, we are thinking to exploit dictionaries, such as *WordNet*, that provide relationships between words. We think that given a term in the query. we could discover its synonyms and use them to rewrite the query (obtaining new neighbour queries).

REFERENCES

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Clark, K. G., Feigenbaum, L., and Torres, E. (2008). Sparql protocol for rdf. *World Wide Web Consortium (W3C) Recommendation*, 86.

Höchtl, J. and Lampoltshammer, T. J. (2016). Adequate-analytics and data enrichment to improve the quality of open data. In *Proceedings of the International Conference for E-Democracy and Open Government CeDEM16*, pages 27–32.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Khosro, S. C., Jabeen, F., Mashwani, S., and Alam, I. (2014). Linked open data: Towards the realization of semantic web - a review. *Indian Journal of Science and Technology*, 7(6):745–764.

Kononenko, O., Baysal, O., Holmes, R., , and Godfrey, M. (2014). Mining modern repositories with elasticsearch. In *MSR. June 29-30 2014, Hyderabad, India.*

Liu, J., Dong, X., and Halevy, A. Y. (2006). Answering structured queries on unstructured data. In *WebDB. 2006, Chicago, Illinois, USA*, volume 6, pages 25–30. Citeseer.

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Miller, E. (1998). An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1):15–19.

Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). Fedx: a federation layer for distributed query processing on linked open data. In *Extended Semantic Web Conference*, pages 481–486. Springer.

Shahi, D. (2015). Apache solr: An introduction. In *Apache Solr*, pages 1–9. Springer.

Winkler, W. E. (1999). The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer.