A Novel Query Language for Data Extraction from Social Networks

Francesco Buccafurri[a], Gianluca Lax[b], Lorenzo Musarella[c] and Roberto Nardone[d]
DIIES, University Mediterranea of Reggio Calabria, Reggio Calabria, Italy

Keywords: Multiple Social Networks, Web Analysis, Query Language, Data Extraction.

Abstract: Online Social Networks (OSNs) represent an important source of information since they manage a huge amount of data that can be used in many different contexts. Moreover, many people create and manage more than one social profile in the different available OSNs. The combination and the extraction of the set of data from contained in OSNs can produce a huge amount of additional information regarding both a single person and the overall society. Consequently, the data extraction from multiple social networks is a topic of growing interest. There are many techniques and technologies for data extraction from a single OSN, but there is a lack of simple query languages which can be used by programmers to retrieve data, correlate resources and integrate results from multiple OSNs. This work describes a novel query language for data extraction from multiple OSNs and the related supporting tool to edit and validate queries. With respect to existing languages, the designed language is general enough to include the variety of resources managed by the different OSNs. Moreover, thanks to the support of the editing environment, the language syntax can be customised by programmers to express searching criteria that are specific for a social network.

1 INTRODUCTION

Over the past decade, Online Social Networks (OSNs) have become widely used by a large set of users in everyday life. Most people create and manage profile in one or more OSNs like Facebook, Twitter, Linkedin, Instagram. Many people spend a lot of time creating many contents on every-day life, latest news, politics, economics, sport, and publish many information (posts, photos, videos, etc.) about their passions, travels, friends. As a consequence, OSNs represent a source of a huge amount of data, offering a great variety of information spanning from personal information to users interests. For this reason, OSNs are recognised as an important phenomenon from a social and economic point of view, and, thus, it is matter of study for the design and the development of innovative and modern web applications. In many cases, both personal information and social interactions coming from social network profiles can be part of innovative software solutions. Among these, social Web applications are the most significant example in which both peoples' identities and contents they produced are involved in the business process and data are mostly owned by users, strongly interlinked and inherently polymorphic (Bell, 2009).

To allow the data extraction, some of the social network providers offer different APIs and framework that reflect their internal metadata. In addition, some other tools and well-known techniques (e.g., web scraping) have been proposed in literature with the aim of extracting data from OSNs. Hence, when a developer needs to interact with different OSNs, she/he needs to manage different techinques and languages. In fact, there is a lack of standard languages and protocols to query both heterogeneous OSNs and multiple OSNs with the same language. In this direction, this paper introduces a novel query language for the extraction of data from multiple OSNs. The proposed language is, according to Martin Glinz's insightful remarks (Glinz, 2002), "as simple as possible and as rich as needed", in the sense that it provides simple keywords and powerful mechanisms to allow writing queries. Moreover, this language offers the possibility to be customized by its users in order to add keywords that better reflect the metadata used by the single OSN.

The paper is organized as follows. Section 2 discusses the related work in the field of data extraction from OSNs. Section 3 describes a data model that is

[a] https://orcid.org/0000-0003-0448-8464
[b] https://orcid.org/0000-0002-5226-0870
[c] https://orcid.org/0000-0002-9616-7256
[d] https://orcid.org/0000-0003-4938-9216

Buccafurri, F., Lax, G., Musarella, L. and Nardone, R.
A Novel Query Language for Data Extraction from Social Networks.
DOI: 10.5220/0008362503650371
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 365-371
ISBN: 978-989-758-386-5

necessary for the formal representation of the multiple social network scenario as a direct graph. Section 4 describes the structure of the query language and its concrete realization in Xtext, also reporting the formal graph. Section 5 reports some usage scenarios in order to clarify the final structure of the queries and to highlight the potentialities of the developed environment. Finally, our conclusions and possible future work are summarized in Section 6.

2 RELATED WORK

As stated previously, some OSNs offer some APIs as a part of their business (Lomborg and Bechmann, 2014). In this way they allow third-party companies to use these APIs to develop complex applications. To the other hand, some works as (Marres and Weltevrede, 2013; Buccafurri et al., 2015) analyzed and exploited web scraping for automated capture of data from OSNs. Web scraping (also known as web harvesting) is a well-known technique to extract online data. This technique is supported by different tools such as Selenium, cURL, Firebug, Node js (Rotruck et al., 1973).

As a consequence, the data extraction and the consequent realization of innovative web applications on the top of social networks is technically feasible. Nevertheless, its complexity is due to different and variable organization of data from the OSNs and the consequent variety of APIs and continuous updates they offer to programmers and/or the different organization of web pages. Indeed, despite the conceptual uniformity among the structures, mechanisms, features of existing social-networks, each platform adopts in practice its own terms, resources, actions. This is a strong handicap for the design and implementation of applications enabling internet-working functions among multiple social networks, and, then, for the achievement of the above goal. As a matter of fact, a few exist in terms of models and languages to support the interaction with multiple social networks.

Up to now a lot of effort has been spent in the definition of standards and models for the data exchange on web. For example, the Resource Description Framework RDF(rdf,) is a W3C standard model for data exchange on web, in particular proposed for Semantic Web. Based on RDF, many researchers proposed different ontologies, because it guarantees interoperability. Indeed, the authors of (Choi et al., 2009) proposed SPIDER, a query processing system for RDF data which supports the SPARQL query language (spa, ; Prud et al., 2006). In the literature, many works use RDF and/or SPARQL approaches for the manipulation and extraction of general data from web sources ((Silva et al., 2017; Diamantini et al., 2018; Thushar and Thilagam, 2008; Barbieri et al., 2010) to cite a few).

Some of the social network providers offer query languages and/or extraction techniques as well. For example, Facebook offers a simple query language (fac, 2008) in order to process mainly your own and your friends' relational data. This language does not give any support for complex data extraction. From 2016, Facebook offers also a specification and a reference implementation of a framework, named GraphQL, which introduces a new type of web-based data access interface (Buna, 2016). This framework is seen as an alternative to the standard REST-based interface (Richardson et al., 2013). The query language at the basis of GraphQL queries uses the JavaScript Object Notation (JSON) (Crockford, 2006) and asks its users to define a schema in JSON. In particular, this schema defines the types of objects and the fields that the query has to populate. Consequently, this approach is very powerful but it is limited to complete knowledge of the possible types and it is also not extensible to multiple-social network scenario. The authors of (San Martın et al., 2011) proposed SNQL, a data model and a query language for Social Network by representing data as graph database. Although the idea and the work are very interesting, these languages cannot be easily extendend to multiple social network scenarios given the different organization of the data.

In the context of multiple social networks, it is more appropriate to analyse the graph query languages. As an example, Neo4J (Miller, 2013) is one of the most complete query languages for Graph Databases, where data can be structured as edges and vertices. GOQL (Sheng et al., 1999) is a different graph query language and it is based on an object-oriented data model. GOQL uses the traditional *select... from... where...* statement for querying and offers also temporal operators such as *next*, *until* and *connected*. The graph query languages can be better adapted to manage complex data structures, where nodes have properties and are organized by relationships, which can also have properties. However, these approaches are not tailored for extracting data from multiple social networks and they are not able to manage the different organization of data from the available sources, which represents, as said before, one of the the main issues in multiple social network.

With respect to the considered works, our approach is different because it aims at developing a complete framework for the creation, verification and execution of queries that are able to extract data from

multiple OSNs. This framework has to offer the following functionalities:

1. Enable users to define customized keywords and the mapping towards the specific metadata adopted by the OSNs;
2. Support to users during the query editing and validate the results;
3. Enable users to execute queries by chosing among the usage of APIs, of the web scraping techniques or of mixed approaches.

With respect to existing languages, the designed language aims at being general enough to include the variety of resources managed by the different social networks. Moreover, it allows users to extend its syntax, adding keywords to the language that map specific metadata offered by a single social network. The design of this language started from the formalization of a conceptual data model for the generalization of the structure of commonly known social networks. Then we defined the syntax and we realized the corresponding grammar file into the Xtext framework (Eysholdt and Behrens, 2010).

3 THE BASIC DATA MODEL

Each Online Social Network has been designed for a particular (main) purpose: for example, Facebook for keeping in touch with friends, Twitter for microblogging, LinkedIn is business-oriented, Instagram for sharing pictures. However, all social networks have some common properties on which they are built: some examples are (1) user's profile, which stores personal data of the user, (2) the concept of relationship, which is necessary to build the connections among users, (3) the presence of shared resources, which can be short texts, Web links, images, videos, and so on, and (4) actions that can be done from users, such as appreciating, commenting, sharing of resources. It is worth noting that social networks usually implement in a different way (or use a different name for) similar properties: for example, the friendship is symmetric in Facebook, whereas it is asymmetric in Twitter, it is built by the concept of *circle* in Google+, it is called *connection* in LinkedIn whereas in Twitter the terms *followers* and *following* are adopted. Again, appreciation is implemented by *like* in Facebook, *endorsement* in LinkedIn, *+1* in Google+ (recently closed).

From these examples, it is clear that, in spite of the intrinsic similarity of social networks, each one defines its own terminology, thus making social network depending on the reference to an entity when we want to write a high-level language for querying social network data: for example, to refer to the contacts of a social network user, we should ask for *connections* if the user is on LinkedIn and *followers/following* if the user is on Twitter. Consequently, the need to have an ontology describing and integrating the characteristics of social networks arises.

In this section, we solve this problem by presenting the ontology used to model social network concepts at an abstract level. The data model presented in this section is general enough and it is independent of the specific social network.

Among the several approaches proposed in the literature to represent social networks, we used an ontology based on the model presented in (Buccafurri et al., 2016), which has been designed with the goal of integrating information coming from different social networks.

Social network data are modeled by a direct graph $G = \langle N, A \rangle$, in which the set of nodes is $N = P \cup R \cup B$, with $P \cap R \cap B = \{\}$, and the set of arcs is $A = F \cup M \cup Pu \cup S \cup T \cup Re \cup L \cup Co$ with $F \cap M \cap Pu \cap S \cap T \cap Re \cap L \cap Co = \{\}$. Now, let's define the sets introduced above.

Each element of the set P represents the *profile* of a user and consists in the tuple ⟨`url, socialNetwork, screen-name, [personalInformation], [picture]`⟩, where `url` is the Web address that identifies and localizes the profile, `socialNetwork` is the commercial name of the Online Social Network which the profile belongs to (the same value is shared by profiles in the same social network), `screen-name` is the name chosen by the user who registered the profile to appear in the home-page of the profile or when posting a resource, and, finally, `personalInformation` and `picture` are the information and the personal image which the user inserted as related to the profile.

The set R models *resources* of the Web or created by users. A resource is represented by a tuple ⟨`url, type, [description], [date]`⟩, where `url` is the Web address to access the resource, `type` indicates the type of the resource content, and finally, `description` and `date` represent the string inserted by who published the resource and the publishing date, respectively.

The set B models *bundles*, a set of resources handled simultaneously by a user, and represented by a tuple ⟨`uri, [description], [date]`⟩, where `uri` is the identifier of the bundle, `description` is the string chosen by the user to be shown with those resources and, finally, `date` represents the publishing date.

The *follow* arcs set $F \subseteq A = \{p_s, p_t \mid p_s, p_t \in P\}$

models the fact that in the (source) profile p_s, it has been declared a certain type of relationship towards the (target) profile p_t.

The *me* arcs set $M \subseteq A = \{p_s, p_t \mid p_s, p_t \in P\}$ denotes that the user with profile p_s has declared in this profile to have a second profile p_t.

The *publishing* arcs set $Pu \subseteq A = \{p_s, b_t \mid p_s \in P, b_t \in B\}$ indicates that the user with profile p_s has published in this profile a bundle b_t.

The *shared* arcs set $S \subseteq A = \{b_s, b_t \mid b_s, b_t \in B\}$ specifies that the bundle b_s (published by a user) is derived from an already published bundle b_t.

The *tagging* arcs set $T \subseteq A = \{p_s, br_t, w \mid p_s \in P, br_t \in B \cup R$ and w is a word$\}$, denotes that the user with profile p_s assigned the word w to describe a bundle or a resource br.

The *referencing* arcs set $Re \subseteq A = \{b_s, p_t \mid b_s \in B, p_t \in P\}$ models the fact that a bundle b_s includes a reference to the profile p_t.

The *like* arcs set $L \subseteq A = \{p_s, pbr_t \mid p_s \in P, pbr_t \in B \cup R \cup P\}$ describes the information that a user with the profile p_s expressed a preference/appreciation for a bundle, a resource or another user profile pbr_t.

The *containing* arcs set $Co \subseteq A = \{b_s, r_t \mid b_s \in B, r_t \in R\}$ indicates that a bundle b_s contains the resource r_t.

4 QUERY LANGUAGE FOR DATA EXTRACTION

Starting from the data model described in the previous section, we define the syntax of the query language which can be adopted by users to easily extract data from multiple social networks. We describe its syntax and its realization through a formal grammar. The structure of our query language is based on the well-known `select` statement of the SQL language, that retrieves zero or more tuples from one or more tables in a database. The basic idea behind this language is to conceptually substitute entities to tuples and social networks to tables in the standard meaning of the `select` statement. It means that, after the `select` keyword, the user has to specify the kind of entity he/she is looking for (i.e., profiles, resources or bundles), then he/she specifies the list of social networks to query after the `from` keyword and, at last, it is possible to add filtering criteria after the `where` keyword. Obviously, the language allows for the use of logical operators in the `where` clause in order to enable more complex predicates.

The final structure of our query language is reported in the following listing.

```
select E1, E2 ...
from S1, S2 ...
where P1 lop P2 ...
```

In the previous listing, `Ei` represents the kind of the requested entity. According to the data model described in the previous section, it is a keyword among `profile`, `resource` and `bundle`. Note that the query language allows selecting different kinds of entities in a single query. `Si` is a conventional name of the social network; the user specifies the set of social networks separating their names by commas. At last, `Pi` is a logical proposition which can be used to filter results; the logical proposition are combined together by logical operators `lop`. Following the data model of the previous section, in the *where* clause the user can specify the value of some elements of the tuple representative of the requesting entity (specified after the *select* keyword). For example, if the user is querying for resources, he/she can specify an url, a type, a description and/or a date. Similarly, if he/she is asking for profiles, it is possible to specify the url, the screen name, personal information and/or picture name as search criteria.

Among the different searching criteria that can be used in the *where* clause, some of them are strictly related to the specific social network. For example, the kind of a resource (e.g., film, book, song) strictly depends on the metadata used by the social network itself which stores that information. For this reason, our language implements a property we named *awareness*. It represents the capability of the language to be extended in keywords by users that are "aware" of the metadata used by every single social network to store its data. Consequently, we can distinguish between "known" and "unknown" social network. In the first category, the user inserts the social networks he/she previously analyzed, while in the second he/she puts the rest of social network he/she wants to use.

The list of social networks the user wants to use is stored in a configuration file. In this file, the user specifies the list of keywords identifying social networks he/she wants to add to the query language in the `from` clause. If a social network is "unknown" (i.e., the user does not know the metadata related to the social network), the user is automatically limited by the language syntax to use a set of general keywords given by the data model; for each "known" social network, instead, he/she has to define a specific configuration file in which he/she details the (sub-)kinds of profiles, resources and bundles can be requested. It means that, if a social network, for example, is able to manage books, films, and songs as kinds of resources, the user is allowed also to ask for a specific kind of them. In this case, the user specifies in the configu-

ration file of the social network the keywords he/she want to use in its concrete language and the mapping with the metadata of the social network. After this one-time operation, he/she can open the language editor and use the keywords he/she specified.

The *awareness* property allows also for future easy integration of additional social networks. The operations needed are: (1) edit the configuration file of the social network and add the new social to the list, (2) possibly create a configuration file for the new social network with the mapping between keywords and metadata.

The described language has been completely realized by means of the Xtext framework (Eysholdt and Behrens, 2010). Xtext is a complete framework for developing programming languages and domain-specific languages with textual grammars. Within Xtext it is possible to define your own language using a powerful grammar language. As a result, you get automatically the full infrastructure including parser, type checker and the entire editing environment for Eclipse.

To realize our query language in Xtext, it is necessary to code the syntax of the language. It has to be defined as a set of rules to which the text has to be compliant. The main rule, named *Query* checks for the overall structure of the query. The following listing reports this grammar rule.

```
Query:
  'select'  selection+=Selection
  'from'
     social+=Social(','social+=
            Social)*
  ('where'   filter+=Filter)? ;
```

This rule specifies the sequence of keywords and textual portions that have to be satisfied by a query. The syntax of each clause has been designed with a specific rule. In the following listing, we report the remaining part of the grammar.

```
Selection:
  (('profile' profile+=Profile(','
     profile+=Profile)*) |
  ('resource'
     resource+=Resource(','
     resource+=Resource)*) |
  ('bundle' bundle+=Bundle(','
     bundle+=Bundle)*))* ;

Social:
  name=ID ;

Filter:
  (condition+=Condition
     (('and'|'or')condition+=
   Condition)*) ;

Condition:
  ((profile+=Profile)? &
  (resource+=Resource)? &
  (bundle+=Bundle)?) ;

Profile:
  ('profileName' name=ID)?
  (('personalInformation'
     personalInformation= STRING)?
     &
  ('screenName'
     ScreenName=STRING)? &
  ('URLprofile' URL=STRING)?) ;

Resource:
  (('type' type+=Type)? &
  ('descriptionResource'
     description=STRING)? &
  ('creation_date'
     creation_date+=Date)? &
  ('URIresource' URI=STRING)?) ;

Type:
  name=ID ;

Bundle:
  (('descriptionBundle'
     description=STRING)? &
  ('creation_date'
     creation_date+=Date)? &
  ('URIbundle' URI=STRING)?) ;

Date:
  d=Day'-'m=Month'-'y=Year ;

Day : INT ;
Month : INT;
Year : INT ;
```

The *Selection* rule mainly checks the usage of the three keywords (profile, resource, and entity), and properly calls three specific rules. The *Social* rule is very simple as, at grammar level, it could check only for the presence of a sequence of social network identifiers. As previously said, the keywords identifying social networks are defined by the user itself in a proper configuration file. At last, the *Filter* rule, together with an additional rule named *Condition*, checks the format of the search filter and allows to create complex combinations of conditions by using logical operators. The filtering mechanisms the user

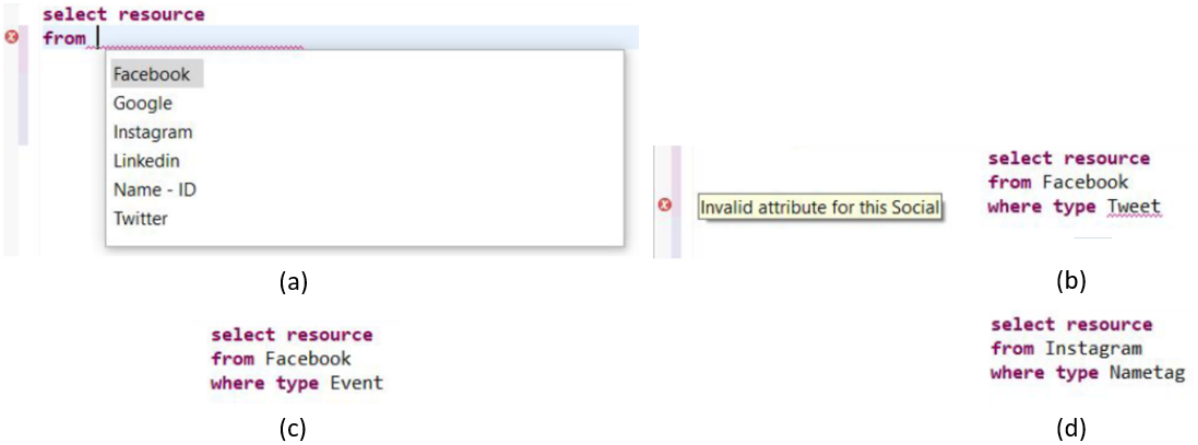

Figure 1: Examples of usage of our query language: (a) content assist enabled; (b) wrong query (c) query for Facebook; (d) query for Instagram.

can specify for entities are defined in the three rules *Profile*, *Resource*, and *Bundle*. These rules check for the presence of keywords related to the attributes of these entities which are given by the data model of the previous section.

The implementation of the *awareness* property of the language, that is the verification of additional keywords in configuration files, is realized by specifying customized validation rules. Specifically, we defined the behavior of the *Validator* class that is used in the Xtext framework to customize the verification activity. Mainly, in this class we inserted the code necessary to check for the presence of the social network identifier in the configuration file and the verification of the existence of the configuration file related to the social network addressed in the query. In this case, the validation allows using additional keywords as given by the configuration file and previously described. The same logic has been implemented also in the *ProposalProvider* class to enable the content assist and suggest users the accepted keywords with respect to the structure of the query.

5 USING THE QUERY LANGUAGE

This section shows some examples of use in order to better clarify the objective of the defined query language and to demonstrate its effectiveness.

Fig. 1 depicts 4 different usage scenario. Fig. 1(a) shows the suggestions of the content assist in the *from* clause. In this scenario, the editing environment retrieved the list of social network from a configuration file and suggest them to a user. The user can select a social network from the shown list, avoiding syntactical errors.

Fig. 1(b) shows a validation error. In this scenario the user specifies that he/she is looking for resources from the Facebook social network and, in the *where* clause he/she filters result on the type *Tweet*. In this case, the environment check that this type (expressed as a keyword) is not specified in the configuration file related to the Facebook and raises an error.

Fig. 1(c) and Fig. 1(d) show two correct usage examples. In the former example the user asks for *nametags* in Instagram, while the latter shows a query requesting *events* from Facebook. These last examples show the concrete support the environment offers to users by enabling the writing of query which uses specific keywords related to the social network metadata.

6 CONCLUSIONS AND FUTURE WORK

This paper proposes a preliminary version of a novel query language for the data extraction from Online Social Networks. The main feature of this language is the possibility to retrieve data from multiple Online Social Networks through a single query. With respect to existing languages, the proposal also offers the possibility for users to add keywords in the language which reflect the metadata used by social networks to address its data. This feature, which we called *awareness*, enables the possibility to add knowledge into the language. The proposed language has been realised by means of the Xtext framework, a well-known framework for the realisation of textual Domain Specific Languages.

This work represents a first step in a wide project which will conduct to the complete development of a framework for data extraction from multiple social networks. In fact, future works include the development of the needed transformations that could automatically translate the queries into executable code, whose execution performs the needed operations on the OSNs to obtain data. We plan to integrate different existing techniques for the data extraction from web sources (such as web scraping and APIs call) in order to design an efficient and effective tool. Of course, the most critical issue is to cope with the continuous changes in web presentation layers and in APIs, offered by the OSNs. At last, we plan to build innovative applications which could use the social data to enforce the trust associated with web profiles.

ACKNOWLEDGEMENTS

This work has been partially supported by the project "Secure Citizen Remote Identification (SECRI)" (CUP J88C17000150006), funded by Regione Calabria - POR Calabria FESR-FSE 2014-2020 - Asse I

REFERENCES

Resource Description Framework . https://www.w3.org/rdf.

SPARQL Query Language for RDF . https://www.w3.org/tr/rdf-sparql-query/.

(2008). Facebook Query Language. http://developers.facebook.com.

Barbieri, D. F., Braga, D., Ceri, S., VALLE, E. D., and Grossniklaus, M. (2010). C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 4(01):3–25.

Bell, G. (2009). *Building social Web applications: Establishing community at the heart of your site*. " O'Reilly Media, Inc.".

Buccafurri, F., Lax, G., Nicolazzo, S., and Nocera, A. (2016). A model to support design and development of multiple-social-network applications. *Information Sciences*, 331:99–119.

Buccafurri, F., Lax, G., Nocera, A., and Ursino, D. (2015). A system for extracting structural information from social network accounts. *Software: Practice and Experience*, 45(9):1251–1275.

Buna, S. (2016). *Learning GraphQL and Relay*. Packt Publishing Ltd.

Choi, H., Son, J., Cho, Y., Sung, M. K., and Chung, Y. D. (2009). Spider: a system for scalable, parallel/distributed evaluation of large-scale rdf data. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2087–2088. ACM.

Crockford, D. (2006). The application/json media type for javascript object notation (json). Technical report.

Diamantini, C., Lo Giudice, P., Musarella, L., Potena, D., Storti, E., and Ursino, D. (2018). An approach to extracting thematic views from highly heterogeneous sources of a data lake. In *Proceedings of the 26th Italian Symposium on Advanced Database Systems (SEBD)*.

Eysholdt, M. and Behrens, H. (2010). Xtext: implement your language faster than the quick and dirty way. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 307–309. ACM.

Glinz, M. (2002). Statecharts for requirements specification-as simple as possible, as rich as needed. In *Proceedings of the ICSE2002 workshop on scenarios and state machines: models, algorithms, and tools*.

Lomborg, S. and Bechmann, A. (2014). Using apis for data collection on social media. *The Information Society*, 30(4):256–265.

Marres, N. and Weltevrede, E. (2013). Scraping the social? issues in live social research. *Journal of cultural economy*, 6(3):313–335.

Miller, J. J. (2013). Graph database applications and concepts with neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, volume 2324.

Prud, E., Seaborne, A., et al. (2006). Sparql query language for rdf.

Richardson, L., Amundsen, M., and Ruby, S. (2013). *RESTful Web APIs: Services for a Changing World*. " O'Reilly Media, Inc.".

Rotruck, J. T., Pope, A. L., Ganther, H. E., Swanson, A., Hafeman, D. G., and Hoekstra, W. (1973). Selenium: biochemical role as a component of glutathione peroxidase. *Science*, 179(4073):588–590.

San Martın, M., Gutierrez, C., and Wood, P. T. (2011). Snql: A social networks query and transformation language. *cities*, 5:r5.

Sheng, L., Ozsoyoglu, Z. M., and Ozsoyoglu, G. (1999). A graph query language and its query processing. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 572–581. IEEE.

Silva, R. R. C., Leal, B. C., Brito, F. T., Vidal, V. M., and Machado, J. C. (2017). A differentially private approach for querying rdf data of social networks. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, pages 74–81. ACM.

Thushar, A. and Thilagam, P. S. (2008). An rdf approach for discovering the relevant semantic associations in a social network. In *Proceedings of the 16th International Conference on Advanced Computing and Communications (ADCOM)*, pages 214–220. IEEE.