An Ontology-based Representation of the Twitter REST API

Konstantinos Togias
School of Science and Technology
Hellenic Open University
Patras, Greece
ktogias@eap.gr

Achilles Kameas
School of Science and Technology
Hellenic Open University
Patras, Greece
kameas@eap.gr

Abstract— **Social Networking Services (SNS) provide users with functionalities for developing their on line social networks, connecting with other users, sharing and consuming content. While most of popular SNS provide open Web 2.0 APIs, they remain disconnected from each other thus fragmenting user's data, social network and content. Semantic social web technologies such as public vocabularies and ontologies can be used for bridging the semantic gap between different SNS. Ontology-based representations of SNS APIs can help developers share knowledge about SNS APIs and can be used for linking APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition. An ontology based representation has been proposed for representing the API of the popular SNS Google+. In this paper, we study the API of Twitter SNS and create an ontology based representation of its structural and functional properties. The proposed Twitter REST API ontology reuses classes of the existing Google+ API ontology and describes valuable structural and functional details of the API, in a machine processable format useful for understanding the API and appropriate for integrating into ontology based Mashups.**

Social Networking System; Web Mashup; Social Semantic Web

I. Introduction

Social Networking Services (SNS) are web applications that allow users create and maintain an online network of close friends or business associates [1]. Typical examples of SNS are Facebook, Myspace, Google+ and Twitter. While SNS have much common functionality they do not usually interoperate and therefore require the user to re-enter her profile and redefine her connections when registering for each service [1]. Also content shared in one SNS is not available to users of other SNS.

Web 2.0 is a widely-used term characterizing the modern web made popular by Tim O' Reilly. Web 2.0 is the network as platform, spanning all connected devices [2]. Web 2.0 applications consume data and services from other applications and enable the reuse and remixing of their own data and services through public Application Programming Interfaces (APIs). Experienced users and programmers use those APIs for creating new integrated web applications, popular known as mashups [3] that combine different data sources and APIs into an integrated end user experience.

Most SNS participate to the Web 2.0 ecosystem by providing their own open APIs. Those APIs provide a first step towards bringing down the walls between SNS. Nevertheless, every SNS use its own terms for defining concepts and representing resources, while it interconnects the resources it provides in its own custom way. Thus common concepts, resources and functionalities are described and provided in different ways in each SNS API.

The Social Semantic Web is the vision of a Web where all of the different collaborative systems and SNS, are connected together through the addition of semantics, allowing people to traverse across these different types of systems, reusing and porting their data between systems as required [1]. Social Semantic Web uses Semantic Web technologies in order to describe in an interoperable way users' profiles, social connections and content creation, sharing and tagging accross different SNS and Sites in the Web.

Ontologies have become the means of choice for knowledge representation in recent years as they provide common format and understanding on domain concepts, while being machine processable [4]. Hendler [5] supports that the ontology languages of the Semantic Web can lead directly to more powerful agent-based approaches. Furthermore, ontologies are used for representing and sharing knowledge about structural and behavioral properties of software [6], for building context-aware and pervasive applications [7], and for achieving context-aware web service discovery and automatic service composition in Service Oriented Software (SOA) [8][9].

Web 2.0 APIs, SOA technologies and Social Semantic Web approaches provide the basic means for bridging the gap between today’s SNS and for unifying users' data, social networks and interactions scattered across various SNS. However, most of today’s SNS APIs lack semantic representations, while existing Semantic Web Ontologies and Vocabularies do not provide links with the API resources and methods used for actually accessing and manipulating users, social networks and content within SNS. Thus, Social Semantic Web approaches, SOA service discovery and service composition techniques cannot be directly applied on them. Moreover, combining multiple SNS APIs for building Mashups require for developers to search, read and combine information from miscellaneous documentation pages scattered across the web. Using Ontologies for describing those APIs can help addressing those shortcomings by providing common, machine processable representations suitable for both sharing knowledge between developers and achieving automatic service discovery and service composition in SNS Mashups. An ontology based representation for the Google+ API has been proposed in [10].

In this work, we study the REST API provided by Twitter, one of the most popular SNS and we propose an ontology based representation of its structural and functional characteristics. Our ontology reuses and extends the work presented in [10], is compatible with the technologies of the Semantic Web and aims to be useful for sharing knowledge about the Twitter REST API between developers of Web 2.0 Mashups and as part of future inter-operable ontology based social networking software.

The paper is organized as follows. Section 2 briefly reviews related work in the areas of Social Semantic Web, Web 2.0 Mashups, ontology representation of software properties, and Service Oriented Architectures (SOA). Section 3 presents the proposed ontology-based representation of Twitter REST API. Section 4 discusses the representation and visualization of the ontology, while Section 5 presents test queries run on the proposed ontology. Section 6 presents conclusions and suggestions for future work.

II. Related Work

Berslin and Decker [11] and Berslin et al. [1] propose the use of Semantic Web mechanisms in order to bridge the isolation and fragmentation of todays SNS. Public vocabularies and ontologies can be used to give meaning to Social Networks and interconnect social websites. The FOAF ontology [12] provides a formal, machine readable representation of user profiles and friendship networks. The SIOC Core Ontology provides the main concepts and properties required to describe information from online communities (e.g. message boards, wikis, weblogs, etc.) on the Semantic Web [13]. The SIOC and FOAF ontologies are used in combination with metadata vocabularies like Dublin Core [14] and SKOS [15] for describing user-generated content on the Social Web. Zhou and Wu in [16] propose an ontology representing SNSs based on FOAF in order to resolve the problem of social data inconsistency and to achieve interoperability among multiple social network services. Their ontology defines some of the basic attributes of a generic SNS API, such as operations, arguments and responses, combined with some user profile and contact attributes borrowed by FOAF ontology, but it does not provide any structural description of the resources that can be accessed through it. While the above approaches describe generic concepts about people, content and SNS, they do not describe the functional and structural aspects of specific SNS APIs necessary for building ontology based Mashups.

The Google+ API Ontology presented in [10] describes the structural and functional properties of the Google+ API. While the Google+ API ontology provides specific instances describing the resources and actions provided by Google+ API, it also defines generic classes describing APIs, types of APIs, authorization protocols, actions, types of resources, parameters, data formats, fields and value types. Those concepts are also useful for describing the Twitter REST API, as it also involves resource types, actions, parameters and values.

Hartmann et al. [17], Zang et al. [3], and Wong and Hong [18] investigate how users with programming skills and programmers build Mashups that make use of public APIs provided by popular web 2.0 services. Most of those users are self-taught and depend on the documentation of the API they want to use. Some of the most common problems encountered when creating Mashups is the complexity of communicating data from one server to another and the lack of proper tutorials and examples in the documentation [3].

Dietrich and Elgar [6] propose that knowledge about structural and behavioural properties of software can be shared across the software engineering community in the form of design patterns expressed in the web ontology language (OWL). The inherent advantage of their approach is that it yields descriptions that are machine processable, but also suitable for a community to share knowledge taking advantage of the decentralized infrastructure of the Internet [6]. Ontology-based representations of SNS APIs can bring the same advantages for the community of Mashup developers.

Kurkovsky, Strimple and Nuzzi in [19] discuss the possibility of convergence of Web 2.0 and SOA, while Xiao et al [8][9] propose the use of ontologies for context-aware web service discovery and automatic service composition. The availability of ontology-based representations of SNS APIs can also help to build software able to automatically compose services that integrate data and functionality from SNS.

Our work takes into consideration and the above works by reusing and extending the classes defined in Google+ API ontology, in order to provide an ontology-based representation of Twitter REST API, compatible with Semantic Web mechanisms and ontology based service discovery and composition approaches that can be used for knowledge sharing and as part of ontology-based Mashups that integrate Twitter functionality and data.

III. An Ontology Based Representation of the Twitter REST API

Twitter is a popular online SNS and microblogging service that enables its users to send and read text-based posts of up to 140 characters, known as "tweets". The service was launched on July 2006. It rapidly gained worldwide popularity, with over 140 million active users as of 2012 generating over 340 million tweets daily and handling over 1.6 billion search queries per day [20] [21].

Twitter REST API follows a RESTful API design, meaning that applications use standard HTTP methods to retrieve and manipulate Twitter resources. Many API calls require that the user of the application is granted permission to access their data. Twitter uses the OAuth 2.0 [22] protocol to allow authorized applications to access user data. Resources in the Twitter REST API can be represented using JSON, XML, RSS, or ATOM data formats. It also supports pagination. The API provides HTTP GET and POST methods for reading and creating, updating or destroying resources. The authors of the documentation of Twitter REST API have grouped the methods provided by the API into 20 main categories: Timelines, Tweets, Search, Streaming, Direct Messages, Friends & Followers, Users, Suggested Users, Favorites, Lists, Accounts, Notification, Saved Searches, Places & Geo, Trends, Block, Spam Reporting, OAuth, Help, Legal, Deprecated. Twitter also provides free client libraries for various programming languages including Python, PHP, Ruby, Javascript and Java.

In order to describe the structural and functional properties of Twitter REST API in a way that can be shared among Software Developers and automatically interpreted by software components, we have introduced an ontology based representation of its main characteristics, resources and actions.

For designing our ontology we followed the steps described by Noy and MacGuinness in [23]:

A. *Specification of the domain and the purpose of the ontology*

The domain of the ontology is the Twitter API and more specifically its structural and functional properties. That is, the data interchange and authentication methods it uses, the types of entities that can be accessed through it and their attributes, and the actions that can be performed through it on these entities. The purpose of the ontology is dual: On the one hand the ontology is playing the role of a shareable and browsable knowledge base for researchers and programmers that want to develop applications and Mashups that integrate Twitter data and functionality, while on the other hand, because of its machine interpretable format, it may be used for building inter-operable ontology based social networking software. Such software will be programmed in a higher level of abstraction and use automatic reasoning on ontologies for providing integration with Twitter.

B. *Enumeration of important terms in the ontology*

For enumerating the important terms in the ontology we studied the Twitter REST API documentation available online [24]. Through the documentation pages we identified references to key terms such as "Authorization", "Field", "Parameter", "Response Format", "Method", "HTTP Method" and "Response Object".

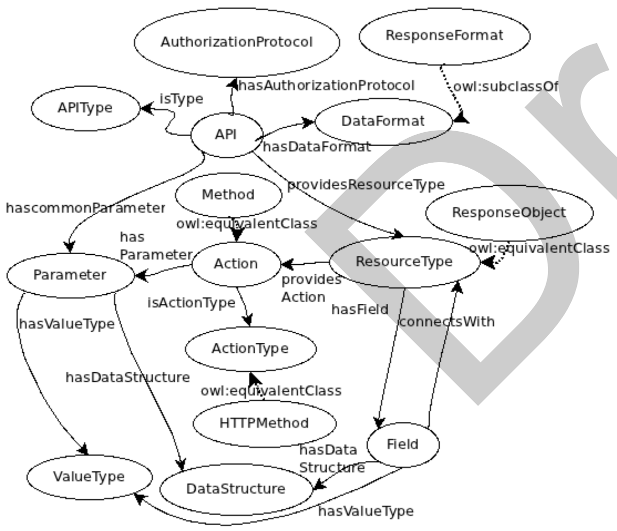


Figure 1. The classes and class hierarchy of Twitter REST API ontology.

C. *Considering reusing existing ontologies*

The Google+ API ontology proposed in [10] provides classes and properties for describing the structural and functional properties of a RESTfull API. The classes API, Parameter, Field, ValueType, DataStructure, AuthorizationProtocol and APIType, defined in Google+ API ontology can also be used for describing the Twitter REST API. Other important terms identified in Twitter REST API documentation pages, such as "*Response Format*", "*Method*", "*HTTP Method*" and "*Response Object*" can be described by *DataFormat*, *Action*, *ActionType* and *ResourceType* classes of the Google+ API ontology. So we decided to reuse and extend the classes defined in Google+ API ontology for building Twitter REST API ontology.

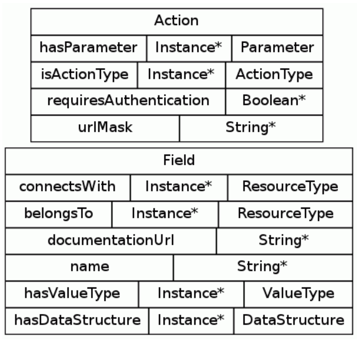
D. *Specification of the classes of the ontology and class hierarchy*

As we stated above, we reused the classes of the ontology defined in [10]. In order to adapt our ontology to the terminology used in the Twitter REST API we defined the following new classes: *ResponseFormat* is a subclass of *DataFormat*, *Method* class is equivalent to class *Action*, *HTTPMethod* class is equivalent to class *ActionType* and *ResponseObject* class is equivalent to class *ResourceType*. Thus, our ontology provides the following classes: *API* (an API), *APIType* (an API type), *DataFormat* (a data interchange format), *ResponseFormat* (a data interchange format used in responses sent by the API), *AuthorizationProtocol* (an authorization protocol used to access the API), *ResourceType* (a resource type provided by the API), *ResponceObject* (equivalent to ResourceType), *Field* (a field of a resource; fields represent attributes of a resource), *Action* (an action that can be performed to Resource), *Method* (equivalent to Action), *ActionType* (an action type), *HTTPMethod* (equivalent to ActionType), *Parameter* (a parameter of an action), *ValueType* (the type of the value contained in a field or a parameter) and *DataStructure* (the type of the data structure contained in a field or a parameter).

E. *Specification of the properties of the classes*

At this step we found that the analysis performed in [10] for Google+ API ontology is also applicable for describing Twitter REST API. Google+ API ontology defines the *connectsWith* object property of *Field* Class for describing the connection between resource types through their fields. Such connections where also identified from the study of the Twitter REST API.

This type of connections is not clearly presented in the Twitter REST API documentation, and a developer has to study the detailed documentation of the responses of various actions in order to detect it. Thus, we re-used the *connectsWith* object property and all the other properties defined in Google+ API ontology for Twitter REST API ontology. Figure 1 depicts the classes and object properties of the Twitter REST API ontology.



Action		
hasParameter	Instance*	Parameter
isActionType	Instance*	ActionType
requiresAuthentication	Boolean*	
urlMask	String*	

Field		
connectsWith	Instance*	ResourceType
belongsTo	Instance*	ResourceType
documentationUrl	String*	
name	String*	
hasValueType	Instance*	ValueType
hasDataStructure	Instance*	DataStructure

Figure 2. Properties and value types of Action and Field classes.

F. *Specification of the value types and restrictions of the properties*

The analysis made in [10] was also applicable in our ontology, so we used the same value types and restrictions. Figure 2 lists the properties and their value types for the *API* and *ResourceType* classes.

G. *Creation of instances*

We derived the Instances of the ontology from the documentation of the API. We represented the Twitter REST API with an instance of the *API class*. Since Twitter REST API is a Restful API, we re-used the *RestfullAPI* instance of the *APIType class* defined in Google+ API Ontology. The API can encode responses in JSON, XML, RSS and ATOM formats, so in addition to the JSON instance of *DataFormat* class defined in Google+ API Ontology we created instances for XML, RSS and ATOM. The API also uses the OAuth authentication protocol for granting access to applications, so we reused the *OAuth* instance of the *AuthorizationProtocol* Class. Twitter REST API provides HTTP GET methods for reading data and HTTP POST methods for writing, updating or deleting data. Thus, in addition to the *GET* instance of *ActionType* Class, we also defined an instance named *POST*, for representing HTTP POST action types.

After studying the parameters and return values of the Actions provided by the API, we found out that in addition to the 5 instances of the *ValueType* class identified in [10], that are also applicable for describing Twitter REST API, the API also uses decimal numbers as values of fields. So, the identified instances of the *ValueType* class are the following 6: *String*, *UnsignedInteger*, *Boolean*, *DateTime*, *ResourceType*, and *Decimal*.

The two instances of the *DataStructure* class, named *SingleValue* and *List*, identified in Google+ API Ontology where also re-used in Twitter REST API ontology.

The documentation of the API explicitly specifies four main response objects (*Tweets*, *Users*, *Entities* and *Places*), but with a more thorough study we identified a much larger number of resource types. Some of them can be directly accessed through actions provided by the API, while other can be accessed through the fields of other resource types. In our ontology we defined all the identified resource types as instances of *ResourceType* Class. Thus we created 44 instances of the *ResourceType* class described in Table 1.

TABLE I. INSTANCES OF THE RECOURCETYPE CLASS

ResourceType Instance	**Description**
Connections	Connections of the authenticating user to other users
DirectMessage	A private message sent from a user to another user
DirectMessages	Direct messages sent to the authenticating user
DirectMessagesSent	Direct messages sent from the authenticating user
Entity	Metadata and additional contextual information about content posted on Twitter
EntityHashTags	Hashtags which have been parsed out of a Tweet text
EntityMedia	Media elements uploaded with a Tweet
EntityUrls	URLs included in the text of a tweet
EntityUserMentions	Other Twitter users mentioned in the text of a Tweet
Favorites	Favorite statuses
FollowersIds	Numeric IDs of users that follow the authenticating user
FriendsIds	Numeric IDs of users that the authenticating user is following
FriendshipsIncoming	Users that have a pending request to follow the authenticating user
FriendshipsLookup	The connections of the authenticating user to other users
FriendshipsNoRetweetIds	The users that the authenticating user does not want to see retweets from
FrienshipsOutgoing	Users for whom the authenticating user has a pending follow request
Help	The current configuration options used by Twitter
HomeTimeline	Statuses, including retweets if they exist, posted by the user and the user's they follow
Language	A language supported by Twitter
Legal	Twitter legal documents
TwitterList	A collection of tweets, posted by users belonging to a curated list
Mentions	Tweets mentioning the user
Place	A geographical place
PublicTimeline	List of statuses, including retweets if they exist, from non-protected users
Relationship	The connection between 2 users
RelationshipSource	The user that is the subject of a relationship
RelationshipTarget	The user that is the target of a relationship
RetweetedByMe	Retweets posted by the authenticating user
RetweetedByUser	Retweets posted by a user
RetweetedToMe	Retweets posted by users the authenticating user follows
RetweetedToUser	Retweets posted by users a user follows
RetweetsOfMe	Tweets of the authenticating user that have been retweeted by others
SavedSearch	A saved search query
Search	Tweets that match a specified query
Status	A tweet
Trend	A popular topic in Twitter
User	A user of Twitter
UserCategory	Categeory of users
UserTimeline	List of tweets posted by the authenticating user
UsersContributees	Users that the specified user can contribute to
UsersContributors	Users that can contribute to the specified user
UsersLookup	Extended information about users
UsersSearch	List of users similar to that returned by the “find people” button on Twitter.com
UsersSuggestions	List of suggested user categories

Finally, we created an instance of Field class for every property of every *ResourceType*, an instance of Action class for every action presented in the documentation of the API, and an instance of the *Parameter* class for every action parameter.

IV. IV. Representation and visualization of the ontology

For the representation of the ontology we used the RDF/XML exchange syntax for the OWL ontology language. We used VIM text editor for editing the XML expressions of the classes and the properties and the specialized ontology editing software Protégé for checking the ontology, creating instances, and producing visualizations.

Figure 3 is a visualization depicting the connections detected between *User* and *Status* resource types in the ontology. From this visualization we observe for example that a resource of type User can be a follower or a friend of another User resource. A User can also be the user (i.e. owner) or retweeter of a Status. Moreover, a Status can reply to a User.

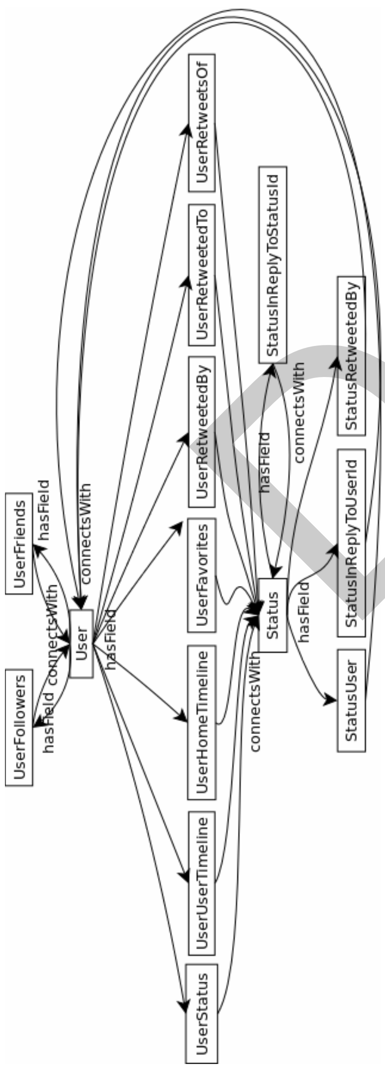


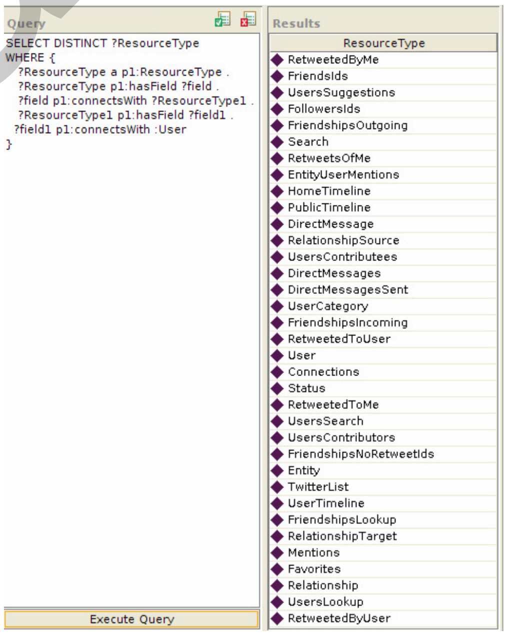
Figure 3. Connections between between User and Status resource types in the ontology.

V. Test queries

In order to test the proposed ontology we run test queries regarding the completeness and correctness of the resulting ontology and validated the results. We queried for all class instances and their properties and cross-checked the returned results with the API documentation pages. We also made sure that all the identified instances were returned.

We also the run two sets of usage test queries and verified the returned results. For the first set of queries, we tried to extract information useful for developers that wish to use the API for building Mashups. Such queries are: (1) What authentication protocol is supported by Twitter API? (2) What is the API's documentation url? (3) What actions and what parameters can be used for directly accessing a User resource? (4) What resources can be directly accessed through the API? (5) What are the resource types that provide a second rank reference to the User resource type (i.e. Have a field that connects to a resource type that has a field that connects to User)?

For the second set of queries we assumed that the ontology is used in ontology-based software for automatically invoking API's methods. Such software needs to extract low-level information about the actual method calls needed for performing an action and the structure of the data needed to be exchanged. Some example queries of this type are the following: (1) What is the APIs base url? (2) What are the data formats supported by the API? (3) What is the urlMask of an Action? (4) What fields are contained in a User resource type and what value type and data structure is each of them?



```
SELECT DISTINCT ?ResourceType
WHERE {
  ?ResourceType a p1:ResourceType .
  ?ResourceType p1:hasField ?field .
  ?field p1:connectsWith ?ResourceType1 .
  ?ResourceType1 p1:hasField ?field1 .
 ?field1 p1:connectsWith :User
}
```

ResourceType
RetweetedByMe
FriendsIds
UsersSuggestions
FollowersIds
FriendshipsOutgoing
Search
RetweetsOfMe
EntityUserMentions
HomeTimeline
PublicTimeline
DirectMessage
RelationshipSource
UsersContributees
DirectMessages
DirectMessagesSent
UserCategory
FriendshipsIncoming
RetweetedToUser
User
Connections
Status
RetweetedToMe
UsersSearch
UsersContributors
FriendshipsNoRetweetIds
Entity
TwitterList
UserTimeline
FriendshipsLookup
RelationshipTarget
Mentions
Favorites
Relationship
UsersLookup
RetweetedByUser

Figure 4. SPARQL query for getting all the resource types that provide second rank access to User resource type.

Moreover if such software is programmed in a higher level of abstraction, it may execute complex queries on the ontology in order to combine data form multiple API resources or to translate generic actions into sequences of API calls. For example: (1) What resource types that can be directly accessed through a GET Action provide a reference to an Status resource type? (2) What POST Actions are provided by TwitterList resource type?

We expressed the above queries in the SPARQL ontology querying language and executed using Protege. Figure 4 depicts a usage test query and the returned results.

VI. Conclusions and Future Work

Ontology-based representations of SNS APIs can help developers comprehend the structure and functionalities of SNS and their APIs and share this knowledge. Moreover they can be used to link those APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition.

In this work we studied the REST API provided by Twitter SNS and created an ontology based representation of its structural and functional properties. For designing the ontology we followed the methodology proposed by Noy and MacGuinness in [23]. We re-used and extended the classes and properties of the Google+ API ontology [10] for building the Twitter REST API ontology. We tested the resulting ontology with SPARQL queries. The proposed ontology reveals the existence of important resources and connections between them that are not clearly presented in the official documentation. We identified a total of 44 resource types in Twitter REST API connecting with each other in various ways. We have made the ontology publicly accessible in OWL format at http://goo.gl/YSbFb.

In this work, we focused on representation of the basic structural and functional features of Twitter REST API such as the resources it provides, the way they connect with each other and the actions they provide. We would like to extend the ontology with descriptions of the authentication process, the manipulation of paging and partial queries and bindings of the actions to client libraries method calls, in order to support automatic invocation of the API calls from ontology driven applications. Moreover, we plan to explore ontology evolution techniques for updating the ontology on the release of API updates. In the near future we would also like to connect the ontology with the Google+ API Ontology [10] and other ontologies and vocabularies like FOAF and SIOC that describe more abstract concepts about users, social networks and content. Finally, we would like to create ontology based representations for the APIs provided by other popular SNS such as Facebook and LinkedIn and to use them for building ontology-based mashups that automatically combine data and functionalities from multiple SNS.

Acknowledgment

This research described in this paper was partly funded by the National Strategic Reference Framework programme 2007-2013, project MIS 296121"Hellenic Open University".

References

[1] J.G. Breslin, A. Passant, and S. Decker , "The Social Semantic Web", Springer-Verlang Berlin Heidelberg, 2009

[2] Tim O'Reilly, "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software", Published in: International Journal of Digital Economics No. 65, March 2007, pp. 17-37.

[3] N. Zang, M.B. Rosson, and V. Nasser, "Mashups: who? What? Why?", In: CHI 2008: CHI 2008 extended abstracts on Human factors in computing systems, ACM, New York, 2008, pp. 3171-3176.

[4] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", In International Journal of Human-Computer Studies, Vol 43 Issue 5-6, Nov./Dec. 1995, pp. 907-928.

[5] J. Hendler, "Agents and the Semantic Web", In IEEE Intelligent Systems, Vol. 16 No 2, 2001, pp. 30-37.

[6] J. Dietrich and C. Elgar, "Towards a web of patterns", In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, num. 2, Elsevier, 2011.

[7] B. Guo, D. Zhang, and M. Imai, "Toward a cooperative programming framework for context-aware applications", In Personal and Ubiquitous Computing, Vol 15, Issue 3, March 2011, pp. 221-233.

[8] H. Xiao et al, "An automatic approach for ontology-driven service composition", Proc. IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 2009, Taipei, Taiwan, 14-15 December 2009, pp 1-8.

[9] H. Xiao et al, "An Approach for Context-Aware Service Discovery and Recommendation", Proc., IEEE International Conference on Web Services (ICWS), 5-10 July 2010, Miami, FL, 2010, pp. 163 – 170.

[10] K. Togias and A. Kameas, "An Ontology-based reperesentation of the Google+ API", Proc., The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services MOPAS 2012, Chamonix Mont-Blanc, May 2012, pp 15-20.

[11] J. Berslin and S. Decker, "The Future of Social Networks on the Internet: The Need for Semantics", IEEE Internet Computing, vol. 11, November 2007, pp. 86-90.

[12] The Friend of a Friend (FOAF) project, online at http://goo.gl/Rdpja, retrieved July 2012.

[13] U. Bojārs and J.G. Breslin (editors), "SIOC Core Ontology Specification", W3C Member Submission 12 June 2007, online at http://goo.gl/8OQV1, 2007, retrieved July 2012.

[14] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set", Version 1.1, online at http://goo.gl/MHLlw, 2010, retrieved July 2012.

[15] A. Miles and S. Bechhofer (editors), "SKOS Simple Knowledge Organization System Reference", W3C Recommendation 18 August 2009, online at http://goo.gl/ypDOU, 2009, retrieved July 2012.

[16] B. Zhou and C. Wu, "Social networking interoperability through extended FOAF vocabulary and service", Proc. 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), 23-25 June 2010, Chengdu, China, 2010, pp. 50 – 55.

[17] B. Hartman, S. Doorley, and S.R. Klemmer, "Hacking, Mashing, Gluing: Understanding Opportunistic Design", in IEEE Pervasive Computing, vol. 7 issue 3, July 2008.

[18] J. Wong, J. and J. Hong, "What do we "mashup" when we make mashups?", Proc. WEUSE '08: Proceedings of the 4th international workshop on End-user software engineering, 2008.

[19] S. Kurkovsky, D. Strimple, and E. Nuzzi, "Convergence of Web 2.0 and SOA: Taking Advantage of Web Services to Implement a Multimodal Social Networking System", proc. 11th IEEE International Conference on Computational Science and Engineering - Workshops, 2008, pp. 227-232.

[20] Wikipedia, Twitter, online at http://goo.gl/MQ3g, retrieved July 2012.

[21] Twitter Search Team (2011-05-31). "The Engineering Behind Twitter's New Search Experience". Twitter Engineering Blog (blog of Twitter Engineering Division), online at http://goo.gl/Cgdly, retrieved July 2012.

[22] E. Hammer-Lahav (editor), "The OAuth 1.0 Protocol, Internet Engineering Task Force (IETF)", online at http://goo.gl/eN6VT, April 2010, retrieved July 2012.

[23] N. F. Noy and D. L. McGuinness, "Ontology development 101: a guide to creating your first ontology". Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford Knowledge Systems Laboratory. Available at http://goo.gl/kr6n4, 2001, retrieved July 2012.

[24] Twitter, Inc., "Twitter REST API", online at http://goo.gl/4NKb3, retrieved July 2