Exploration of Semi-Structured Data Sources

Thiago Nunes, Daniel Schwabe

Department of Informatics
Pontifical Catholic University of Rio de Janeiro
R.M.S. Vicente 225
Gávea Rio de Janeiro, RJ, Brazil
+55 21 3527-1500

{tnunes, dschwabe}@inf.puc-rio.br

Abstract. There has been a large growth of available semi-structured data on the Web, spurred both by governmental requirements for publishing public data, and by private sector, for various purposes. One such large initiative is the Linked Open Data Cloud. An increasingly important activity is to make sense of such published data, often exploring it as a prelude or as initial steps to perform some information-processing task. Exploration is then a generalization of the traditional search task, as it involves other operations beyond finding specific information. The design and evaluation of exploratory frameworks is a complex, multi-disciplinary endeavor, with important challenges for both aspects. In this paper, we will argue the need to separate the conceptual exploratory operations users may carry out over semi-structured data from the particular interface designs used to give users access to such operations. We illustrate the problems using practical examples and state-of-the-art tools and discuss how this separation of concerns allows more accurate evaluation of the relevant aspects of any proposed tool or framework that aims at supporting Explorations.

1 Introduction

The exploration of (semi) structured data is a highly interdisciplinary research area [23] where the goal is learning something from successive data manipulation and cognitive activities [15, 18, 22, 24, 25]. It covers aspects that range from algorithmic issues of the information retrieval system, dealing with Human-Computer Interaction (HCI) aspects, and visualization techniques. The exploration phenomenon is frequently referred as "Exploratory Search" in the literature, a term introduced by Marchionini [15] in 2006.

Exploratory Search is usually considered a process that combines searching and browsing activities aiming at knowledge acquisition [24]. However, in our vision, the exploration of semi-structured information goes beyond searching and browsing, involving also management of the knowledge acquired along the process, as well as reuse and sharing of exploration solutions, preferably leveraged by a formal exploration model. For this reason we refer to the process of exploration of (semi) structured datasets as *Information Exploration*.

Despite the attempts to identify the various concerns in Information Exploration, such as operations, interaction patterns and visualizations [6, 23, 24], the evaluation of the exploration tools does not consider or discuss the influences of each aspect in isolation. Analyzing the evaluations of the exploration tools, we observe that the results in general support the hypothesis of the presumed benefits, but lack proper assessment of both the outcomes and of the exploration process. It is also hard to figure out the range of tasks for which the tools are more suitable since the authors usually use as measures the task completion and learnability of interface mechanisms [8, 10, 18] but don't discuss interaction dialogue structures or the available functions and their applications to solve exploration problems.

In this work we will shed some light on how to adopt a pragmatic model-driven separation of concerns in order to characterize the information exploration tools by both the set of operations they provide and the physical interface dialogue structure that support the execution of those operations.

2 Different Concerns in Information Exploration

Although the separation of user tasks, operations and goals from interface design details has been widely recognized as valuable approach in HCI since the existence of task models, such as the Goals, Operations, Methods and Selection rules (GOMS) family [14], it has not been applied in the context of exploration tools. One of the consequences is the difficulty to assess both to which range of exploration tasks the tools are suitable and how well they support the user during an exploration task.

The separation of concerns in information exploration proposed in this paper is consistent with Norman's theory of gulf traversal [17], which separates the user-system interaction in two major phases: the "execution gulf" and the "evaluation gulf". The "execution gulf" covers all the way starting with the user's intention of executing an action and ends with the translation of this intention in terms of interface controls. The "evaluation gulf" concerns the interpretation and assessment of the results generated by the "execution gulf". The semantic and articulatory distances govern the gulf traversals. While the semantic distance is the distance between the user's intention and the actual system operations set, the articulatory distance stands between the meaning of those operations and the physical means to execute them through the system interface. In this work, we propose the assessment of the semantic distance concern of exploration tools through the available exploration functions set. The articulatory distance is assessed through the user-system interaction dialogue structure required to execute those operations.

2.1 Exploration Functions

As mentioned previously, the user's intentions and actions at the cognitive level can be captured as exploration functions. Our research is based on Pirolli's levels of explanation of the user's interaction with information [18]. As an example, consider the cognitive functions of *Pivoting* and *Querying*. *Pivoting* is an action that allows the user to change the context or the focus of exploration, e.g. a user changing the analysis of a musical artist to one of his compositions. We define *Querying* as the action of specifying the characteristics of the desired items to be retrieved by the system. Similar to *Pivoting* and *Querying*, there are other functions that capture the user's intended exploration actions. The complete description of the whole set of exploration functions is beyond the scope of this paper, but still we briefly describe here the most common:

- *KeywordSearch(Keywords)*: perform a search for the occurrence of the provided keywords over the dataset and returns the set of items matching the keywords;
- *Pivot*: changes the focus of exploration to another related item, or adds a pivot to the current focus in case of multi-pivoting exploration. In the simplest case, the *Pivot* operation just puts another related element as the pivot of exploration in a one-to-one style similar to the traditional hyperlink navigation between web pages (for example, *American President* to *birth place*). In more advanced forms, *Pivot* can express many-to-many pivoting from a set of items to another related set of items – *Pivot(Items, relation)* –, e.g., *Pivot(AmericanPresidents, wife)* returns the set of American President's wives;
- *Query(Characteristics)*: retrieves information items through the specification of their characteristics. For example, consider a user retrieving all theaters in London by specifying the type of the element and the location: Query(type:Theater, location:London);
- *Project(Items,Characteristic)*: projects some set of results along a property. For example, projecting a set of works by the year of publication to be plotted over a line chart: *Project(Works, publicationYear)*;
- *GroupBy(Items, Characteristic)*: groups a result set based on the values of some characteristic of the information items. As an example, consider a user grouping European companies by their areas of expertise;
- *Refine(Items, Filters)*: refines a set of items through the application of filters received as parameters, e.g., Refine(Publications, {year > 2004});
- *FindPath:* finds structural connections between sets of information items [1, 11, 19] . Consider a user trying to find how a company "A" is related to company "B", for instance, because they actually share a field of expertise. *FindPath(CompanyA, CompanyB)* can be used to discovery such hidden relationship.

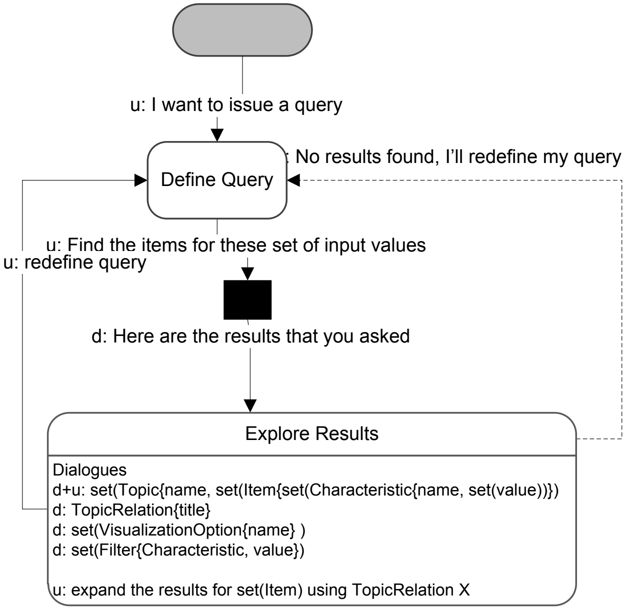
We can model the process as functional compositions by considering the exploration process as a sequence of function applications over a dataset, where the output of one function is used as input of the next,. The extent of possible functional compositions that is supported by some exploration tool depends on the set of primitive functions it implements, ultimately determining its expressivity to support Information Exploration. We expect that comparisons addressing expressivity issues would shed some light on how well the exploration tools assist the user during the task execution as well as the types of tasks that are better supported. The more expressive is the set of exploration functions for a given task, the shorter is the semantic distance to bridge in both the execution and the evaluation gulfs.

2.2 Exploration Functions Vs. Interaction Patterns

In this section we will illustrate and discuss how the same functions can be articulated differently in different tools, and how we can capture those differences in a conversational model for qualitative comparisons. For this discussion we selected the *Pivot* and *Query* functions and show how they can be composed through several interaction patterns. To illustrate our point, we examined two state-of-the-art tools – Liquid Query [7] and SeCo tool [6] – and specified interaction diagrams using the Modeling Language for Interaction as Conversation (MoLIC) model [3]. Fig. 1 and Fig. 2 show the diagrams for the *Pivot* and *Query* exploration functions for Liquid Query and SeCo.

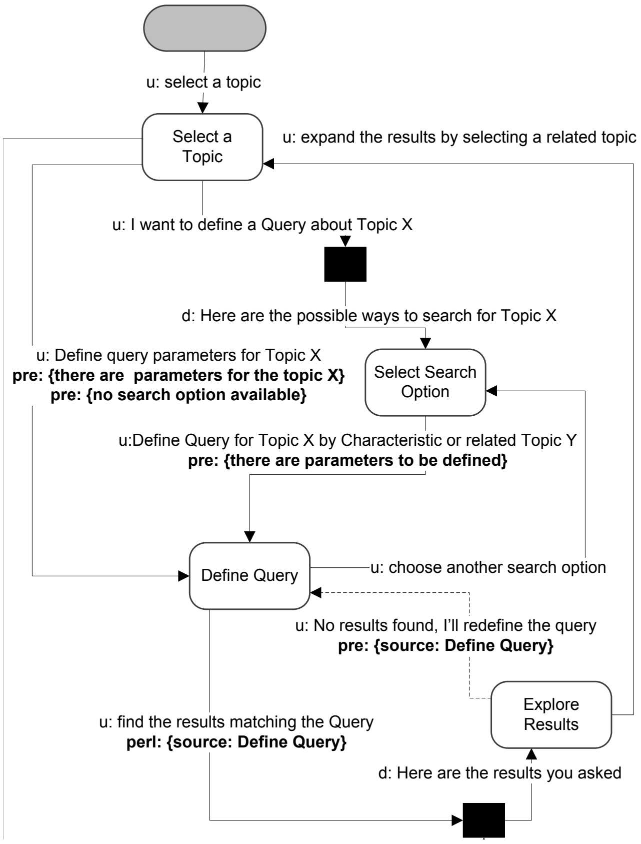
MoLIC interaction diagrams are organized around dialogues between the user and the system, considered as the designer's deputy, and *dialogue scenes*, which represent user-system dialogues about a certain topic. There are also *transition utterances*, which represent the turn taking in the conversation. The scenes are represented as white named boxes. The user-system utterances are represented as labeled arrows and usually cause scene transitions. Utterances emitted from the user have "u:" indication while the designer's deputy utterances have a "d:" indication. The black boxes represent a system processing step. Inside the scene boxes the information items involved in the user-system dialogue are presented.

Fig. 1. Liquid Query dialogue structure for Query and Pivot functions.


u: I want to issue a query
Define Query
: No results found, I'll redefine my query
u: Find the items for these set of input values
u: redefine query
d: Here are the results that you asked
Explore Results
Dialogues
d+u: set(Topic{name, set(Item{set(Characteristic{name, set(value))}})
d: TopicRelation{title}
d: set(VisualizationOption{name})
d: set(Filter{Characteristic, value})
u: expand the results for set(Item) using TopicRelation X

The interaction with both tools starts through some ubiquitous access (gray ellipses). In Liquid Query (Fig. 1) the user first executes the Query in the "Define Query" scene. After that, the user can *Pivot* in the "Explore Results" scene by asking the designer's deputy to expand the result set using some topic relationship between sets of information items (utterance "*u: expand the results for set(Item) using TopicRelation X"* in Fig. 1). In SeCo, the execution of the *Query* function in the "Define Query" scene can only be achieved before the user selects a topic of search ("Select a Topic" scene in Fig. 2) and how the information items will be queried ("Select Search Option" scene in Fig. 2). The *Pivot* function in SeCo is also revealed through a different interaction pattern. In SeCo the user can ask the system to pivot through a result set expansion (utterance "*u: expand the results by selecting a related topic*" in Fig. 2). This solicitation causes a scene transition that leads the user-system dialogue to the topic selection and search option scenes. From this example, we conclude that both the exploration functions and the way they can be composed can assume different interaction patterns that clearly influence the articulatory distance to bridge the gulfs.

Fig. 2. SeCo dialogue structure for Query and Pivot functions


u: select a topic
Select a Topic
u: expand the results by selecting a related topic
u: I want to define a Query about Topic X
d: Here are the possible ways to search for Topic X
u: Define query parameters for Topic X
pre: {there are parameters for the topic X}
pre: {no search option available}
Select Search Option
u:Define Query for Topic X by Characteristic or related Topic Y
pre: {there are parameters to be defined}
Define Query
u: choose another search option
u: No results found, I'll redefine the query
pre: {source: Define Query}
Explore Results
u: find the results matching the Query
perl: {source: Define Query}
d: Here are the results you asked

Looking only at the *Pivot* function we can observe that it may be defined in different ways, such as one-to-one [5], where the user pivots from one item to another (from the artist to one composition), one-to-many (from the artist to the set of his/her compositions), and many-to-many [2, 13, 19] (from the set of Brazilian composers to the set of their compositions). Hence, a designer should address the design of the *Pivot* function both at the cognitive and semantic levels by deciding whether it receives and returns single or multiple items, and at the articulatory level by deciding which interaction pattern will be adopted for both the concrete execution of the action and the composition with the ensuing exploration functions. These design decisions should be guided by the environment and tasks for which the system will be used, as well as by the target users' profile and background.

2.3 Exploration Case: Patent Analysis

Patent Analysis is a kind of activity that can often be characterized as an exploration task [9] according to the characteristics described in [25]. Patent analysis can be carried out for different purposes, such as granting intellectual property rights to an applicant, describing tendencies of technological advancement in a specific area, mining relationships of a company and its competitors, or tracing the profile of companies with regards to technological innovation investments. In order to accomplish these tasks, patent analysts usually have to analyze manually hundreds of patents retrieved by a patent database query [16]. Therefore, we choose patent analysis tasks for our use case due to the high demand on the set of exploration functions in order to aid the performance of the analysts.

Consider the following scenario:

"Company X is aiming to invest in a different area of its current investments in order to diversify its activities. A promising area of investment is the development of Lithium-ion traction batteries. However, Company X lacks knowledge in the area and decided to hire a patent analyst to trace a profile of the main players in the area in order to better understand it. The patent analyst should prepare a report containing the following information:

- *Trace and compare the activity of each company in the last 10 years.*
- *What are other areas of investment in addition to Lithium-ion Traction Batteries that the players are addressing? What are the activities in those areas?"*

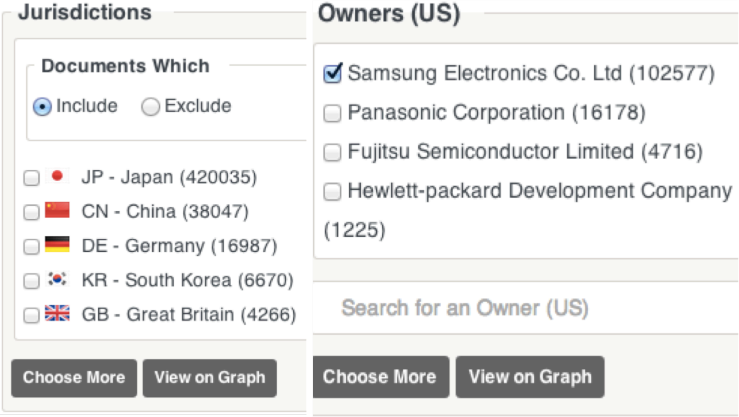
In order to analyze how well is the support of an exploration tool for the execution of the scenario, we selected one of the state-of-the-art tools for patent exploration, the Patent Lens system[1]. First we simulate the ideal execution in Patent Lens and then we show how we can improve the process by augmenting the expressivity of its set of functions.

Patent Lens is a faceted system with many visualization options, such as line charts, pie charts, and bar charts. The facets available for exploration, among others,

[1] http://www.lens.org/lens/search

are: Date, Jurisdiction, Inventor, Owner, Applicant and Classification. In order to explore the dataset, users have to select one or more values for the facets and refine the current result set. When the user selects one of the facets, the current result set is grouped by the possible values of the selected facet. Fig. 3 shows some possible values for the *Jurisdiction* and *Owner* facets along with the number of results they achieve. We capture the intention of grouping the result set along the facet values in the *GroupBy* exploration function.

Fig. 3. Example of two facets in Patent Lens.



Jurisdictions

Documents Which

● Include ○ Exclude

☐ JP - Japan (420035)

☐ CN - China (38047)

☐ DE - Germany (16987)

☐ KR - South Korea (6670)

☐ GB - Great Britain (4266)

Owners (US)

☑ Samsung Electronics Co. Ltd (102577)

☐ Panasonic Corporation (16178)

☐ Fujitsu Semiconductor Limited (4716)

☐ Hewlett-packard Development Company (1225)

Search for an Owner (US)

In order to solve the first problem of tracing the activity of each player the shortest strategy the user can employ is:

```
1.  KeywordSearch ("Lithium-ion Traction Batteries")
2.  Refine({year > 2004, year <= 2014});
3.  GroupBy(Owner);
4.  For each owner o;
            Project(Refine(Owner = o), Year)
```

As we can observe, it is only possible to project one result set at a time, generating a loop on the set of patent owners. If we extend the expressivity of the *Project* function to receive one or many sets of items to project, the task would have lower costs in terms of number of actions.

Another interesting example is the problem of tracing other areas of investment of the Lithium-ion traction battery players. Using Patent Lens, the shortest execution strategy is:

```
1.  KeywordSearch ("Lithium-ion Traction Batteries")
2.  For each owner in GroupBy(Owner)
            Annotate owner using an external tool
3.  For each class in GroupBy(Classification)
            Annotate class using an external tool
```

```
4. Return to the starting point: All Patents
5. GroupBy(owner)
6. For each annotatedOwner in AnnotatedOwners
       Refine(Owner = annotatedOwner)
7. For each class in GroupBy(Classification)
       If (class not in AnnotatedClasses): Refine(Classification = class)
```

The key strategy to solve this subtask is to find the disjoint classes from the classes of patents related to the keyword “Lithium-ion Traction Batteries”. First the user has to annotate the owners and the classifications using an external tool. After that, the user has to clean all filters and return to the initial state, which contains all patent documents. Next, he has to refine the result set to keep just the patents of the previously annotated Lithium-ion traction battery players since the tool does not feature any function to save and reuse the set of owners found in previous steps. Finally the user has to refine the set for each classification that was not in the set of annotated classes related to Lithium-ion traction batteries. As a conclusion, we can observe that the higher cost actions are the ones that involve loops for each information item. We can minimize the cost by adding some set-based functions to the set of exploration functions of Patent Lens.

If we improve the expressivity of the set of exploration functions by adding *Pivot* and set difference – *Diff* – functions, we can reduce drastically the complexity of finding the disjoint classes by eliminating the loop. Therefore, the actions could have the following structure:

```
1. KeywordSearch (“Lithium Traction Batteries”): R1
2. Pivot (R1, classification) -> R2
3. Pivot (R1, owner) -> R3
4. Pivot(R3, hasPatent) -> R4
5. Pivot(R4, classification) -> R5
6. Diff(R5, R2) -> DisjointClasses
```

The *Pivot* function changes the focus of exploration by generating a set of related information items, such as the set of classifications of patents related to the keywords in step 2. The *Diff* function generates the difference between two sets and is applied in the last step to extract all classes that are not in the set of classifications of patents related to “Lithium-ion Traction Batteries” keyword.

From the examples above, we conclude that it is possible to improve the expressive power of an exploration tool by adding new primitives to the set of exploration functions. Once an adequate functional model is devised for the target exploration tasks and the target users, the system interface should be modeled to aid the translation of the functions in interface controls. In the next section we will show the benefits of modeling the information exploration as a composition of functions.

3 A Functional Model for Information Exploration

In the previous section we have argued for the existence of two layers that are usually addressed indiscriminately: the functional layer and the interaction layer. In the functional layer the users' cognitive exploration actions and strategies can be modeled as functions and functional compositions, respectively. The interaction layer is concerned with the translation of those functions in interface controls. In this section we give more details of the functional layer showing the benefits of addressing it formally and separately from interface design and implementation concerns. We also illustrate how a framework of exploration functions can enable comparisons and evaluations of information exploration tools.

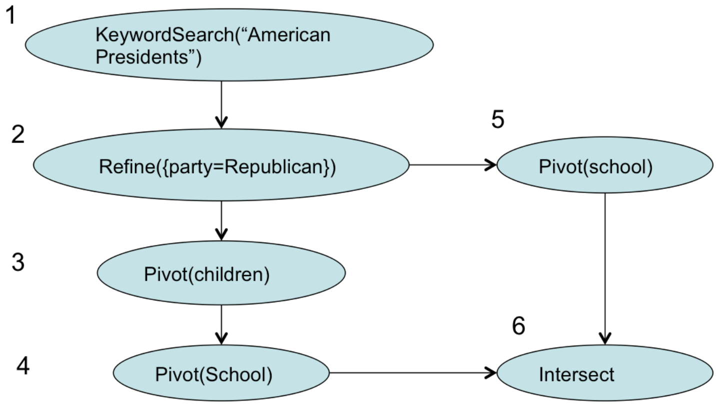
3.1 Evaluation and Comparisons of Exploration Tools

The most frequent problem in the experiments addressing the exploration tools is the inability of assessing the exploration process. It is not possible to assess to what extent and how good is the tool support for exploring information. As an example, we selected different tools to analyze only their functional aspect. For demonstration purposes, consider the following exploration task, which is an extension of the problem presented in [13]:

"Finding Schools of Republican American Presidents' children. Which schools have received both the Presidents and their children?"

The cost of the process required to solve this task can be significantly different depending on the expressivity of the set of exploration primitives available in exploration tools. Fig. 4 shows the steps required to solve the task with Explorator.

Fig. 4. Graphical representation of the functional composition required to solve an exploration task with Explorator


1
KeywordSearch("American Presidents")
2
Refine({party=Republican})
5
Pivot(school)
3
Pivot(children)
6
4
Pivot(School)
Intersect

In Fig. 4 the user starts with a keyword search for the topic "American Presidents" (step 1) and refines the set of presidents by their party (step 2). Next, the user pivots from the set of presidents to the set of presidents' children (step 3). In order to find the schools the presidents' children have attended, the user pivots again using the "School" relationship (step 4). At this step, the user recognizes the need to also add the set of presidents' schools as another pivot (step 5). Finally the schools who have received both presidents and presidents' children is achieved in the step 6 by intersecting the set of schools achieved in step 4 with the set of schools achieved in step 5.

Multi-pivoting can be defined as the possibility of adding distinct sets of elements as the focus of exploration, hence, leveraging operations over multiple pivots, such as finding relations between them [19]. We address the multi-pivoting as a characteristic of the *Pivot* operation. An environment features multi-pivoting when the execution of the *Pivot* operation adds an element, or a set of elements, to the current exploration focus instead of replacing the current focus.

In the execution presented in Fig. 4, while in Explorator the set of primitives includes set operations and multi-pivoting, which allows the user to intersect the results of two pivoting operations in steps 4 and 5, Parallax [13], gfacet [10], and /facet [12] have more restricted expressive power. In Parallax, it is not possible to work with multiple pivots since the *Pivot* operation was designed to replace the current focus of exploration. Therefore, the user has to backtrack from step 4 to step 2 in order to achieve the results of step 5. Moreover, there are no set operations in Parallax, which forces the user to calculate the intersection in step 6 manually. gfacet do allow multi-pivoting, hence, steps 4 and 5 can be achieved without backtracking. However, there are no available operations to process multiple sets of elements, hence, the exploration trail can only assume the format of a tree. Therefore, in gfacet, step 6 also requires a lot of manual effort for large result sets. /facet is even more restricted in terms of expressive power since it is not possible to pivot through a specific relationship, as in steps 4 and 5. It should be noted that the system in [8] enables a form of filtering that also allows avoiding backtracking.

As a conclusion, we showed that it is possible to compare exploration tools just considering the available set of exploration functions. By doing this, we reinforce both the existence of an additional layer that should be considered in the design process independently of interface concerns and the benefits that a formal exploration framework can bring for tool evaluation and comparison concerns.

4 Conclusion and Future Directions

Although the separation of the conceptual model of user operations from the interface details is a widely accepted principle in HCI design, it has hitherto not been properly applied in the context of information exploration tools. Consequently, the evaluations and experiments usually fail in explaining the reasons of observed successes or failures. Moreover, it is difficult to compare these tools regarding the adequacy for information exploration tasks without a common framework of operations. Given this scenario, the contributions of this paper are two fold. First, we demonstrate through use cases how exploration tools can be assessed both in terms of the operations set and in terms of the dialogue structure present in the user interface. Second, we propose new usage of two distinct models for qualitative evaluation and comparison of exploration tools. The first model, which is still in construction, is a framework of exploration operations. The second model is the Modeling Language for Interaction as Conversation, which is an abstraction to analyze tools regarding their conversational structure.

Modeling the user's exploration as a nested sequence of function applications over a dataset allows us to represent the process as functional compositions. We claim that the expressivity of exploration tools can be assessed by the range of functional compositions that can be formed using the set of primitive exploration functions offered. Therefore, a formal framework of exploration operations would leverage the usage of expressivity to more accurately evaluate and compare exploration tools independently of interface design issues. In order to illustrate this position we used examples of real exploration problems and showed how the same exploration functions can be presented with different interaction patterns and how we can improve the exploration process simply by evolving the set of primitive functions.

As future work, we plan to elaborate a formal description of a framework of exploration operations and evaluate how well it leverages the description and representation of the Information Exploration process. We will evaluate and compare exploration tools regarding the expressivity of the set of primitive exploration functions. We also plan to study how the same framework can be used as a formal base for reuse of exploration patterns and knowledge sharing among communities of users.

Acknowledgement

The authors were partially supported by CNPq project 557128/200-9 National Science, Technology Institute on Web Science, CAPES, and Google Research Program.

5 References

1. Araujo, S. et al.: Fusion - Visually Exploring and Eliciting Relationships in Linked Data. Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I. pp. 1–15 Springer-Verlag, Berlin, Heidelberg (2010).
2. Araújo, S. De, Schwabe, D.: Explorator: a tool for exploring RDF data through direct manipulation. Linked data on the web WWW2009 workshop (LDOW2009). (2009).
3. Barbosa, S.D.J., Greco, M. de P.: Designing and Evaluating Interaction as Conversation: A Modeling Language Based on Semiotic Engineering. Interactive Systems. Design, Specification, and Verification. pp. 16–33 Springer Berlin Heidelberg (2003).
4. Bates, M.J.: Information search tactics. J. Am. Soc. Inf. Sci. 30, 4, 205–214 (1979).
5. Berners-lee, T. et al.: Tabulator : Exploring and Analyzing linked data on the Semantic Web. 3rd International Semantic Web User Interaction Workshop. Vol. 2006. (2006).
6. Bozzon, a et al.: Exploratory search framework for Web data sources. VLDB J. 22, 5, 641–663 (2013).

7. Bozzon, A. et al.: Liquid Query : Multi-Domain Exploratory Search on the Web. Proc. 19th Int. Conf. World wide web. 161–170 (2010).
8. Buschbeck, S. et al.: Parallel faceted browsing. CHI ’13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13. p. 3023 ACM Press, New York, New York, USA (2013).
9. Forti, E., Toschi, L.: Patents as Measure of Exploration and Exploitation Strategy : The Case of CVC Investments.
10. Heim, P. et al.: gFacet : A Browser for the Web of Data. International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW’08), Vol. 417, (2008).
11. Heim, P. et al.: Interactive Relationship Discovery via the Semantic Web. Proc. 7th Ext. Semant. Web Conf. (ESWC 2010). 6088, C, 303–317 (2010).
12. Hildebrand, M. et al.: /facet: A Browser for Heterogeneous Semantic Web Repositories Michiel. The Semantic Web-ISWC 2006. Springer Berlin Heidelberg, 272-285. (2006).
13. Huynh, D., Karger, D.: Parallax and companion: Set-based browsing for the data web. WWW Conf. C, (2009).
14. John, B. E., Kieras, D. E.: The GOMS family of user interface analysis techniques: comparison and contrast. ACM Transactions on Computer-Human Interaction, *3*(4), 320–351, (1996), doi:10.1145/235833.236054
15. Marchionini, G.: Exploratory Search: from finding to understanding. Commun. ACM. 49, 4, 41–46 (2006).
16. Mukherjea, S. et al.: Information retrieval and knowledge discovery utilizing a biomedical patent semantic Web. IEEE Trans. Knowl. Data Eng. 17, 8, 2005–2008 (2005).
17. Norman, D.A., Draper, S.W.: User Centered System Design; New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (1986).
18. Pirolli, P.: Information Foraging Theory: Adaptive Interaction with Information. Oxford University Press, Inc., New York, NY, USA (2009).
19. Popov, I. et al.: Connecting the dots: a multi-pivot approach to data exploration. International Semantic Web Conference - ISWC 553–568 (2011).
20. Patient protection and affordable care act. Public Law. 111–148 (2010).
21. De Souza, C.S. et al.: The Semiotic Inspection Method. Proceedings of VII Brazilian Symposium on Human Factors in Computing Systems. pp. 148–157 ACM, New York, NY, USA (2006).
22. Vakkari, P.: Exploratory Searching As Conceptual Exploration. Proc. Symp. Human-Computer Interact. Inf. Retr. - HCIR ’10. 3–6 (2010).
23. White, R.W. et al.: Exploratory search interfaces: Categorization, Clustering and Beyond. ACM SIGIR Forum. 39, 2, 52 (2005).
24. White, R.W., Roth, R.A.: Exploratory Search: Beyond the Query-Response Paradigm. Synth. Lect. Inf. Concepts, Retrieval, Serv. 1, 1, 1–98 (2009).
25. Wildemuth, B.M., Freund, L.: Assigning search tasks designed to elicit exploratory search behaviors. Proc. Symp. Human-Computer Interact. Inf. Retr. - HCIR ’12. C, 1–10 (2012).