Using Knowledge Anchors to Facilitate User Exploration of Data Graphs

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Marwan Al-Tawil[a,*], Vania Dimitrova[a], Dhavalkumar Thakker[b]
[a] *School of Computing, University of Leeds, Leeds, UK*
[b] *School of Electrical Engineering and Computer Science, University of Bradford, Bradford, UK*

Abstract. This paper investigates how to support a user's exploration through a data graph in a way leading to expanding the user's domain knowledge. To be effective, approaches to facilitate exploration of data graphs should take into account the utility from a user's point of view. Our work focuses on *knowledge utility* – how useful exploration paths through a data graph are for expanding the user's knowledge. We propose a new exploration support mechanism underpinned by the subsumption theory for meaningful learning, which postulates that new knowledge is grasped by starting from familiar entities in the data graph which serve as knowledge anchors from where links to new knowledge are made. A core algorithmic component for adopting the subsumption theory for generating exploration paths is the automatic identification of knowledge anchors in a data graph (KA_{DG}). Several metrics for identifying KA_{DG} and the corresponding algorithms for implementation have been developed and evaluated against human cognitive structures. A subsumption algorithm which utilises KA_{DG} for generating exploration paths for knowledge expansion is presented and applied in the context of a data browser in a music domain. The resultant exploration paths are evaluated in a controlled user study to examine whether they increase the users' knowledge as compared to free exploration. The findings show that exploration paths using knowledge anchors and subsumption lead to significantly higher increase in the users' conceptual knowledge. The approach can be adopted in applications providing data graph exploration to facilitate learning and sensemaking of layman users who are not fully familiar with the domain presented in the data graph.

Keywords: Data graphs, knowledge utility, data exploration, meaningful learning, knowledge anchors, exploration paths.

1. Introduction

In the recent years, linked data (in the form of RDF graphs) has emerged as the de facto standard for sharing data on the Web. Consequently, data graphs have become widely available on the Web and are being adopted in a range of user facing applications that provide search and exploration tasks. In contrast to regular search where the user has a specific need in mind and an idea of the expected search result [1], exploratory search is open-ended requiring significant amount of exploration [2], has an unclear information need [3], and is used to conduct learning and investigative tasks [4]. For example, when people explore resources in a new domain (like in academic research tasks) or browse through large information spaces with many options (like exploring job opportunities, travel and accommodation offers, videos, music). In many cases, the users will have no (or limited) familiarity with the specific domain. When the users are novices to a domain, the *users' cognitive structures* about that domain are unlikely to match the complex *knowledge structures of data graphs that represent the domain*. This can have a negative impact on the user exploration experience and effectiveness. Users can find themselves in situations where they are not able to formulate knowledge retrieval queries (users do not know what they do not know [3]). Users can face an overwhelming amount of exploration options, not being able to identify

*Corresponding author (E-mail: altawilmarwan@gmail.com) is a PhD candidate supervised at and by University of Leeds.

which exploration paths are most useful; this can lead to confusion, high cognitive load, frustration and a feeling of being lost.

To overcome these challenges, appropriate ways to facilitate user exploration through data graphs are required. Research on exploration of data graphs has come a long way from initial works on presenting linked data in visual or textual forms [5,6]. Recent studies on data graph exploration have brought together research from diverse, but related, areas such as Semantic Web, personalisation, adaptive hypermedia, and human-computer interaction; with the aim of reducing user cognitive load and providing support for knowledge exploration and discovery [7–9]. Several attempts addressed supporting *layman users*, i.e. users who are novices to the domain. Examples include: personalising the exploration path tailored to the user's interests [10], presenting RDF patterns to give an overview of the domain [11], or providing graph visualisations to support navigation [12].

Although a significant amount of work addresses the problem of facilitating user exploration through data graphs, this has been applied mainly in investigative tasks. There is limited research on supporting learning through data graph exploration. The learning perspective has been studied with regard to providing generic tools for exploration of interlinked open educational resources [13]. We address an important outstanding challenge by focusing on the learning effect of user exploration, i.e. expanding the user's domain knowledge while he/she is exploring a data graph in an unfamiliar or partially familiar domain. Supporting learning through search is an emerging research area in information retrieval [14,15]. It argues that "searching for data on the Web should be considered an area in its own right for future research in the context of search as a learning activity" [16]. Motivated by this vision, we investigate how learning can be supported through exploration of data graphs.

The work presented in this paper opens a new avenue in semantic data exploration, which looks at the *knowledge utility*, i.e. expanding one's domain knowledge while exploring a data graph. This builds on earlier work showing that while exploring data graphs in unfamiliar (or partially familiar) domains, users *serendipitously* learn new things that they were unaware of [17–19]. However, not all exploration paths can be beneficial for knowledge expansion, e.g. paths may not bring new knowledge to the user or may bring too much unfamiliar things so that the user becomes confused and frustrated [18].

The main goal of our research is to develop automatic ways to generate exploration paths that can expand the user's domain knowledge. A scoping user study which investigated several exploration strategies for knowledge expansion [20] pointed us to the adoption of Ausubel's subsumption theory for meaningful learning [21] as the underpinning model for the generation of exploration paths. This theory postulates that human cognitive structures are hierarchically organised with respect to levels of abstraction, generality, and inclusiveness of concepts; hence, familiar and inclusive entities are used as *knowledge anchors* to subsume new knowledge into the users' cognitive structures. Such entities are crucial for expanding one's domain knowledge.

Adopting Ausubel's meaningful learning theory, our work addressed the following research questions:

RQ1. *How to develop automatic ways to identify knowledge anchors in a data graph?* This means finding graph entities which correspond to domain concepts that can be familiar to layman users.

RQ2. *How to use knowledge anchors to generate exploration paths to facilitate domain knowledge expansion?* This means suggesting to the user graph entity sequences for exploration, which can result in learning new things about the domain.

To address ***RQ1***, we utilise Rosch's notion of basic level objects (BLO) [22] in the context of data graphs. We devise a formal framework that maps Rosch's definitions of BLO and cue validity to data graphs, and develop several metrics and the corresponding algorithms to identify knowledge anchors in a data graph (KA_{DG}). These anchors represent *concepts which are likely to be familiar to layman users*, and hence can be used as knowledge bridges of familiar entities from where links to new knowledge can be made. To evaluate the KA_{DG} algorithms, we compare the derived KA_{DG} against human cognitive structures. Using an experimental approach that adapts Cognitive Science methods to derive BLO in a domain, we identify human basic level objects in a data graph (BLO_{DG}) used to benchmark the KA_{DG} algorithms. Based on the evaluation, we identify hybridisation heuristics to improve precision and recall.

To address ***RQ2***, we develop an algorithm that is underpinned by the subsumption theory for meaningful learning [21] to generate exploration paths for knowledge expansion. This involves two steps – identifying the most appropriate knowledge anchor to start the path, and iterative introduction of new knowledge through subsumption to connected entities. We have conducted a controlled task-driven user study with a semantic data browser in the Music do-

main to examine the effectiveness of the suggested paths to increase the users' domain knowledge. The study shows that the generated paths have significantly higher knowledge utility and provide better exploration experience than free exploration paths.

The main contributions of this work are:

- formal description and implementation of metrics and the corresponding algorithms for identifying KA_{DG};
- formal description and implementation of an algorithm for identifying BLO_{DG} which correspond to human cognitive structures over a data graph;
- analysis of the performance of KA_{DG} metrics, and suggested hybridization heuristics, using BLO_{DG} identified by humans applied to a data graph in the Music domain;
- formal description and implementation of an algorithm for generating exploration paths using KA_{DG} by adopting the subsumption theory for meaningful learning; and
- analysis of the knowledge utility and usability of automatically generated exploration paths based on knowledge anchors and subsumption against free exploration of a data graph; using a semantic data browser in the Music domain.

The paper is structured as follows. Section 2 positions the work in the relevant literature and compares to similar approaches. Section 3 presents the research context for experimentation – a semantic data browser and describes its data graph. Section 4 provides preliminaries with key definitions, followed by a formal description of metrics for identifying KA_{DG}, and the corresponding algorithms for applying these metrics over a data graph (Section 5). Sections 6 describes an experimental study to validate the KA_{DG} algorithms against human BLO_{DG}. A subsumption algorithm used to generate exploration paths for knowledge expansion is described in Section 7, while Section 8 presents an experimental study to evaluate the knowledge utility of the generated paths against free exploration. Section 9 discusses the findings, followed by conclusions in Section 10.

2. Related Work

In the context of the research presented in this paper, we review relevant research on data graph exploration approaches to justify the main contributions of this work. We also compare to existing approaches for identifying key entities and generating paths in data graphs. Since our work involves several evaluation steps, we review relevant evaluation approaches.

2.1. Exploration through Data Graphs

Semantic data exploration approaches are divided into two broad categories: (i) visualisation approaches for data graph exploration [23–26] and (ii) text-based semantic data browsers [27–30]. Visualisation provides an important tool for exploration that leverages the human perception and analytical abilities to offer exploration trajectories. These approaches, in addition to intuitiveness, focus on the need for managing the dimensions in semantic data represented as properties, similarity and relatedness of concepts. The text-based browser approaches operate on semantically augmented data (e.g. tagged content) with layout browsing trajectories using relationships in the underpinning ontologies. These are adopting techniques and knowledge from learning, human-computer interaction and personalisation to enhance the data exploration experience of users.

2.1.1. Visualisation Approaches for Data Graph Exploration

The state of the art in approaches that harness visualisation as a tool for exploratory discovery and analysis of linked data graphs is presented in [23]. They use Sheiderman's seminal work on visual information seeking (overview first, zoom and filter, then details-on-demand) [31] to evaluate the usability and utility of these approaches and focus on: (i) how well these approaches generate summary of data; (ii) how well they focus on finding relevant and important data; and (iii) what visualisation techniques are used. In [32] the authors utilise cartographic metaphor and visual information seeking principles to offer overview of data based on instance types, and then automatically generating SPARQL queries based on search and interaction with a map. The focus of their work is the entry point of the vast data graph the users have to explore. There are numerous approaches to support visual SPARQL query construction and many of these works [33–35] have similar target audience, i.e. a layman user who is unfamiliar with the domain of the data graph. These works use visualisation techniques to help layman users browse through large data graphs. For example the work in [33] introduced a graphical interface (called NITELIGHT) for semantic query construction which is based on the specification of SPARQL query language. The interface allows users to create

SPARQL queries using a set of graphical notations and editing action. The state of the art approaches to support visual SPARQL query construction is presented in [36]. The focus of their work is in supporting layman users to perform exploratory querying of RDF graphs in space, time, and theme with interactive visual query construction methods. The authors present a number of design principles to counter the challenges and evaluate them in a usability study on finding maps in a historical map repository [36]. Although these approaches hide complexity of graph terminologies, their primarily focus on helping layman users to generate SPARQL queries instead of focusing on the properties of data graphs to guide user exploration. [37] is such a work that manipulate the data graph properties to guide exploration with a system called Aemoo that helps users to focus on the most important bit of information about an entity first and then explore other related information. Aemoo utilises encyclopaedic knowledge patterns as relevance criteria for selecting, organising, and visualising knowledge. They are discovered by mining the linking structure of Wikipedia to build entity-centric summaries that can be exploited to help the users in exploratory search tasks. However, this approach is feasible in multi-knowledge domains that are built by humans (e.g. Wikipedia) and may not be feasible in specific domains with complex structures. Furthermore, the system considers one level below the root, and does not cover different entities at different level of abstractions.

These visualization efforts are geared towards helping layman users explore the complex graph structures by hiding the complexity of semantic terminology from the users. However, the effectiveness of the visualizations depends on the user's ability to make sense of the graphical representation which in many cases can be rather complex. Users who are new to the domain may struggle to grasp the complexity of the knowledge presented in the visualisation. Our approach to automatically identify entities that are close to the human cognitive structures can be used as complementary to visualisation approaches to *simplify* the data graph by pointing at entities that layman users can be familiar with. The prime focus of our approach is on augmenting text-based data browsers by offering exploration paths.

2.1.2. Text-based Semantic Data Browsers

Two types of linked data browsers had emerged since the early days– (i) pivoting (or set-oriented browsing) browsers and (ii) multi-pivoting browsers. In a pivoting browser, a many-to-many graph browsing technique is used to help a user navigate from a set of instances in the graph through common links [27]. Exploration is often restricted to a single start point in the data and uses 'a resource at a time' to navigate anywhere in a dataset [28]. This form of browsing is also referred as uni-focal browsing. A second type of browsers supports multi-pivoting that allows a user to start from multiple points of interest. For example, PolyZoom enables multi-focus exploration of maps for a user to zoom various parts of the map at the same time [35]. Different multi-pivoting interfaces have also been proposed, including Parallax [29], VisiNav [38], PepeSearch [30].

A noteworthy variation of the pivoting approach is the use of facets for text-based data browsing of linked datasets. Faceted browsing is the main approach for exploratory search in many applications. The approach employs classification and properties features from linked datasets as a mean to offer facets and context of exploration. Facet Graphs [39], gFacet [40], and tFacet [41], are early efforts in this area. More recent attempts include Rhizomer [42] that combines navigation menus and map to provide flexible exploration between different classes, Facete [43] a visualization-based exploration tool that offers faceted filtering functionalities, Hippalus [44] allows users to rank the facets according to preferences defined directly by the user, Voyager [45] is a mixed-initiative system that couples faceted browsing with visualization recommendation to support users exploration, and SynopsViz [46] a tool that offers multi-level visual exploration and analysis over RDF and provides faceted browsing and filtering over classes and properties. Although these approaches provide support for user exploration, layman users who are performing exploratory search tasks to learn or investigate new topic, can be cognitively overloaded especially when facets provide lengthy options (i.e. multiple links) for users to explore. The authors in [11] proposed Sview, a browser that utilises a link pattern-based mechanism for entity-centric exploration over Linked Data. Link patterns describe explicit and implicit relationships between entities and are used to categorise linked entities. A link pattern hierarchy is constructed using Formal Concept Analysis (FCA), and three measures are used to select the top-k patterns from the hierarchy. However, the approach lacked considering the user preference in identifying the link patterns in supporting exploratory search tasks such as learning, which would be challenging especially when the domain is not familiar to the user.

These initial approaches have become more sophisticated, personalising the exploration in order to not only hide the complexity of semantics, but also to take into consideration the user's profile and interests. An approach that explicitly targets personalisation in semantic data exploration using users' interests is presented in [47]. Linked dataset concepts are dynamically categorised into upper mapping and binding exchange layer concepts using a fuzzy retrieval model. Results with the same concepts are grouped together to form categories, later used during concept-based browsing to align the exploration space to the users' interests. Recent approaches aimed to improve search efficiency over Linked Data graphs by considering user interests [48], or to diversify the user exploration paths with recommendations based on the browsing history [49]. A method for personalised access to Linked Data has been suggested in [50] based on collaborative filtering that estimates the similarity between users, and then produces resource recommendations from users with similar tastes. Similarity between users is calculated by taking into account the commonalities, the informativeness and the connectiveness of the shared resources between the users. More recently, a graph-based recommendation methodology based on a personalised PageRank algorithm has been proposed in [51]. This adopts a personalisation vector that assignes different weights to different nodes to get a bias towards nodes closer to user preferences. The personalisation approach in [52] allows the user to rate semantic associations represented as chains of relations that may reveal interesting and unknown connections between different types of entities for personalised.

The above approaches stress the importance of tailoring the exploration to the users. None of them investigates the user's familiarity with the domain, which is the main focus of the approach we present here where familiarity is related to understanding and knowledge expansion. Moreover, conventional personalization approaches suffer from the 'cold start' problem – for a reliable user model to be obtained, the user should have to spend time interacting with the system to provide sufficient information about their interests. Instead, we exploit the structure of the knowledge graph to identify entities that are likely to be familiar to users, which overcomes the cold start problem. Strictly considered our approach is not personalisation, because we do not dynamically adapt to the user's knowledge as it expands while the user browses through the graph. However, knowledge anchors can be seen as a way to approximate what entities in the domain may be familiar to the user which is used in the algorithms we propose for generating exploration paths.

2.2. Identifying Key Entities in Data Graphs

The most recent statistics [1] show that there are 3360 interlinked, heterogeneous datasets containing approximately 3.9 billion facts. The volume and heterogeneity of such datasets makes their processing for the purpose of exploration a daunting challenge. Utilisation of various computational models makes it possible to handle this challenge in order to offer fruitful exploration of semantic data. Finding key entities in a data graph is an important aspect of such computational models and is generally implemented using ontology summarization [53] and formal concept analysis [54] techniques.

Ontology summarisation has been seen as an important technology to help ontology engineers to make sense of an ontology, in order to understand, reuse and build new ontologies [24,55,56]. The process of summarising an ontology involves identifying the key concepts in an ontology, and then hiding/missing concepts that are not key concepts [57]. A good summary should be concise, yet it needs to convey enough information to enable a decent understanding of the original ontology, yet provide an extensive coverage of the entire ontology [58]. Centrality measures have been used in [53] to identify key concepts and produce RDF summaries. The notion of relevance based on the relative cardinality and the in/out degree centrality of a node has been used in [59] to produce graph summaries. The state of the art ontology summarization approach is presented in [58]. The approach exploits the structure and the semantic relationships of a data graph to identify the most important entities using the notion of relevance, and then select edges connecting the entities by maximising either locally or globally the importance of the selected edges. The notion of relevance is based on the relative cardinality (i.e. judging the importance of an entity from the instances it contains) and the in/out degree centrality (i.e. the number and type of the incoming and outgoing edges) of an entity.

The closest ontology summarisation approaches to the context of our work relates to extracting key concepts as the best representatives of ontology [60,61]. It highlights the value of cognitive natural categories for identifying key concepts in an ontology to aid ontology engineers to better understand the ontology and quickly judge the suitability of an ontology in a

[1] http://stats.lod2.eu/

knowledge engineering project. The authors applies a name simplicity approach, which is inspired by the cognitive science notion of basic level objects [22] as a way to filter entities with lengthy labels for the ontology summary. The work in [61] has utilised basic level concepts to extract ontologies from collaborative tags. The work proposes an algorithm for constructing an ontology using basic level concepts. A metric based on the category utility is proposed to identify basic concepts from collaborative tags, where tags of a concept are inherited by its subconcepts and a concept has all instances of its descendants. However, these approaches lack applying the formal definitions of basic level objects and cue validity described in [22,62] in the context of a data graph. Our work has operationalised these definitions by developing several algorithms with the corresponding algorithms for identifying knowledge anchors in a data graph.

Formal Concept Analysis (FCA) is a method for analysis of object-attribute data tables [54], where data is represented as a table describing objects (i.e. taxonomical concepts), attributes and their relationships. FCA has been applied in different application areas [63,64] such as Web mining and ontology engineering. In *Web mining*, FCA based approaches have been used to improve the quality of search results presented to the end users. For example, the work in [65] developed a personalised domain-specific search system that uses logs of keywords and Web pages previously entered visited by other persons to build a concept lattice. More recently, FCA has been applied to construct a link pattern hierarchy to organise semantic links between entities in a data graph [11]. In *ontology engineering*, FCA has been used in two topics: ontology construction and ontology refinement. The work in [66] used FCA to construct ad hoc ontologies to help the user to better understand the research domain. In [67] the authors presented OntoComp, an approach for supporting ontology engineers to check whether an OWL ontology covers all relevant concepts in a domain, and supports the engineers to refine (extend) the ontology with missing concepts. The psychological approaches to basic level concepts have been formally defined for selecting important formal concepts in a concept lattice by considering the cohesion of a formal concept [68]. This measures the pair-wise similarity between the concept's objects based on common attributes. More recently, the work in [69] has reviewed and formalised the main existing psychological approaches to basic level concepts. Five approaches to basic level objects have been formalised with FCA [69]. The approaches utilised the validity of formal concepts to produce informative concepts capable of reducing the user's overload from a large number of concepts supplied to the user.

Existing studies in ontology summarisation and FCA utilise BLO to identify key concepts in an ontology in order to help experts to examine or reengineer the ontology. They have been evaluated with domain experts, and are applicable in tasks where the users have a good understanding of the domain. In contrast, we apply the notion of BLO in a data graph to identify *concepts which are likely to be familiar to users who are not domain experts*. Focusing on layman users, we provide unique contribution that adopts Rosch's seminal cognitive science work [22] to devise algorithms that identify (i) KA_{DG} that represent familiar graph entities, and (ii) BLO_{DG} which correspond to human cognitive structures over a data graph. Crucially, these algorithms are validated with layman users who are not domain experts.

2.3. Generating Paths in Data Graphs

In data graphs, the notion of path queries uses regular expressions queries to indicate start and end entities of paths in data graphs [70]. For example, in a geographical graph database representing neighborhoods (i.e. places) as entities and transport facilities (e.g. Bus, Tram) as edges, the user writes a simple query such as "I need to go from Place *a* to Place *b*", and the user is then provided with different transportations facilities going through different routes (paths) starting from Place *a* to reach the destination Place *b* [70]. Another used notion is *property paths.* A *property path* defines possible routes between any two given entities in a data graph. A trivial case of a property path is simple triple pattern of length 1. Property paths are used to capture associations between entities in data graphs where an association from entity *a* to entity *b* comprises entity labels and edges [71]. However, in big data graphs there are usually high numbers of associations (i.e. possible property paths) between the entities and ways to refine and filter the possible paths, are required. To tackle this challenge, the work in [7] presented Explass[2] for recommending patterns (i.e. paths) between entities in a data graph. A pattern represents a sequence of classes and relationships (edges). Explass uses frequency of a pattern to reflect its relevance to the query. It also uses informativeness of classes and relationships in the pattern to indicate its informa-

[2] http://ws.nju.edu.cn/explass/

tiveness by adding the informativeness of all classes and relationships. A class having fewer instances is more specific and thus more informative. The idea is similar to relationships (i.e. an edge) except that informativeness is considered to the start (*Subject*) entity and target (*Object*) entities of a relationship. Relfinder[3] [72] is another approach for helping users to get an overview of how two entities are associated together by showing all possible paths between two entities in an RDF graph. This approach may not be suitable for layman users, who may become confuse or overloaded with too much unfamiliar entities.

While several approaches address the problem of supporting users' exploration through data graphs, none of them aims at providing layman users with exploration paths to help such users to expand their domain knowledge. Furthermore, earlier approaches generate paths that link graph entities, and are suitable for tasks where users explore associations between known entities. Whereas, we provide paths for uni-focal exploration where the user starts from a single entry point and explores the data graph. The unique feature of our work is the explicit consideration of knowledge utility of exploration paths. We are finding entities that are likely to be familiar to the user and using them as knowledge bridges to gradually introduce unfamiliar entities and facilitate learning. This can enhance the usability of semantic data exploration systems, especially when the users are not domain experts. Therefore, our work can facilitate further adoption of linked data exploration in the learning domain. It can also be useful in other applications to facilitate the exploration by users who are not familiar with the domain presented in the graph.

2.4. Data Exploration Evaluation Approaches

Evaluation of data exploration applications usually considers the exploration utility from a user's point of view or analyses the application's usability and performance (e.g. precision, recall, speed etc.) [23]. The prime focus is assessing the usability of semantic Web applications, while assessing how well the applications help the users with their data exploration tasks is still a key challenge [73]. Task driven user studies have been utilised to assess whether a data exploration application provides useful recommendations for accomplishing users exploration tasks [37]. A task driven benchmark for evaluating semantic data exploration has been presented in [73]. The benchmark presents a set of information-seeking tasks and metrics for measuring the effectiveness of completing the tasks. The evaluation approach in [74] aimed to identify whether the simulated exploration paths over information networks are similar to those produced by human exploration.

In the context of ontology summarisation, there are two main approaches for evaluating a user-driven ontology summary [55]: gold standard evaluation, where the quality of the summary is expressed by its similarity to a manually built ontology by domain experts, or corpus coverage evaluation, in which the quality of the ontology is represented by its appropriateness to cover the topic of a corpus. The evaluation approach used in [60] included identifying a gold standard by asking ontology engineers to select a number of concepts they considered the most representative for summarising an ontology.

In this paper, we evaluate algorithms for identifying knowledge anchors in data graphs by comparing the algorithms' outputs versus a benchmarking set of basic level objects identified by humans. To the best of our knowledge, there are no evaluation approaches that consider key concepts in data graphs which correspond to cognitive structures of users who are not domain experts. Our evaluation approach that identifies BLO_{DG} through an experimental method adapting Cognitive Science methods is novel and can be applied to a range of domains. Our second evaluation study which assess the knowledge utility and usability of the generated exploration paths adopts an established task-based approach and utilises an educational taxonomy for assessing conceptual knowledge.

3. Application Context

As an application context for algorithm validation and user evaluation we use MusicPinta - a semantic data browser in the music domain [18]. MusicPinta provides a uni-focal interface for users to navigate through musical instrument information extracted from various linked datasets. The MusicPinta dataset includes several sources, including DBpedia[4] for musical instruments and artists, extracted using SPARQL CONSTRUCT queries. The DBTune[5] dataset is utilised for music-related structured data. Among the datasets on DBTune.org we utilise: (i) Jamendo which is a large repository of Creative Commons licensed music; (ii) Megatune is an independent music label; and (iii) MusicBrainz is a com-

[3] http://relfinder.dbpedia.org

[4] http://dbpedia.org/About.

[5] http://dbtune.org/.

munity-maintained open source encyclopaedia of music information. The dataset coming from DBTune.org (such as MusicBrainz, Jamendo and Megatunes) already contains the "sameAs" links between them for linking same entities. We utilise the "sameAs" links provided by DBpedia to link MusicBrainz and DBpedia datasets. In this way, DBpedia is linked to the rest of the datasets from DBtune.org, enabling exploration via rich interconnected datasets. The MusicPinta dataset is made available as an open source on sourceforge[6]. All datasets in MusicPinta are available as a linked RDF data graph and the Music ontology[7] is the ontology used as the schema to interlink them.

The dataset has 2.4M entities and 38M triple statements, taking 1.5GB physical space. Table 1 shows the main characteristics of the MusicPinta dataset.

Table 1. Main characteristics of MusicPinta data graph. The data graph includes five class hierarchies. Each class hierarchy has number of classes linked vie the subsumption relationship `rdfs:subClassOf` (e.g. there are 151 classes in the `String Instrument` class hierarchy). DBpedia categories are linked to classes in a hierarchy via the `dcterms:subject` relationship, and classes are linked via the domain-specific relationship `MusicOntology:instrument` to musical performances. The depth of each class hierarchy is the maximum depth value for of the entities in that class hierarchy.

Instrument class hierarchy	No. of classes	Depth	No. of DBpedia categories	No. of music performances
`String`	151	7	255	348
`Wind`	108	7	161	1539
`Percussion`	82	5	182	127
`Electronic`	16	1	7	11
`Other`	7	1	0	2

The music ontology provides sufficient class hierarchy for experimentation. For instance, the class hierarchies for the `String` and `Wind` musical instruments have depth of 7, which is considered ideal for applying the cognitive science notion of basic level objects [22] on data graphs, as this notion states that objects within a hierarchy are classified at least three different levels of abstraction (superordinate, basic, subordinate). The MusicPinta dataset provides an adequate setup since it is fairly large and diverse, yet of manageable size for experimentation.

Figures 1 and 2 show examples of the user interface in the MusicPinta semantic data browser.

[6] http://sourceforge.net/p/pinta/code/38/tree/

[7] http://musicontology.com/.

Fig. 1. Description page of the entity `'Xylophone'` in MusicPinta, extracted from DBpedia using CONSTRUCT queries.

Fig. 2. Semantic Links (i.e. predicates) related to entity `Xylophone` presented in *Features* and *Relevant Information*. *Features* include the semantic relationships `rdf:type` (e.g. `Xylophone` is an instrument), and semantic relationships `rdfs:subclassOf` (e.g. the subClass `Xylophone` belongs to the superClass `Tuned Percussion`) and `dcterms:subject` relationship that links an entity to its DBpedia category (e.g. `Xylophone` belongs to the category `Greek loanwords`). *Relevant Information* also include the semantic relationship `rdfs:subclassOf` (e.g. `Celesta` is a subClassOf `Xylophone`)

In a scoping user study, we examined user exploration of musical instruments in MusicPinta to identify what strategies would lead to paths with high knowledge utility (details of the study are given in [20]). We examined two dimensions - the *user's familiarity with the domain* and the *density of entities in the data graph*. Consequently, three exploration strategies were suggested based on these two dimensions: (i) density-strategy – directing the users to *densest* entities; (ii) familiarity-strategy – asking the user to always select entities that are *familiar* to him/her; and (iii) unfamiliarity-strategy – asking the user to always select entities that are *unfamiliar* to him/her. Paths which included familiar and dense entities and brought unfamiliar entities led to increasing the users' knowledge. For example, when a participant was directed to explore the entity `Guitar` which he/she was familiar with, the participant could

see unfamiliar entities linked to `Guitar` such as `Resonator Guitar` and `Dobro`. The entity `Guitar` served as an anchor from where the user made links to new concepts (`Resonator Guitar` and `Dobro`). We also noted that the dense entities, which had many subclasses and were well-connected in the graph, provided good potential anchors that could serve as bridges to learn new concepts.

Subsumption Theory for Meaningful Learning. While the scoping study provided useful insights, it did not give solid theoretical model for developing an approach that generalises across domains and data graphs. However, the study directed us to Ausubel's subsumption theory for meaningful learning [21] as a suitable theoretical underpinning for generating exploration paths. This theory [21,75–77] has been based on the premise that a human cognitive structure (i.e. individual's organization, stability, and clarity of knowledge in a particular subject matter field) is the main factor that influences the learning and retention of new knowledge [77]. In relation to meaningful learning, the subsumption process postulates that a human cognitive structure is hierarchically organised with respect to levels of abstraction, generality, and inclusiveness of concepts. Highly inclusive concepts in the cognitive structure can be used as *knowledge anchors* to subsume and learn new, less inclusive, sub-concepts through meaningful relationships [21,76,78,79]. Once the knowledge anchors are identified, attention can be directed towards identifying the presentation and sequential arrangement of the new subsumed content [77]. Hence, to subsume new knowledge, anchoring concepts are first introduced to the user, and then used to make links for introducing new concepts.

In the following sections, we will utilise the meaningful learning theory to generate exploration paths through data graphs based on knowledge anchors. We will split this into two stages: (i) identifying knowledge anchors in data graphs, and (ii) using the knowledge anchors to subsume new knowledge.

4. Preliminaries

We provide here the main definitions that will be used in the formal description of the main algorithms.

Linked Data graphs are built using traditional Web standards (e.g. Uniform Resource Identifiers (URIs) and Hypertext Transfer Protocol (HTTP) and use a common data graph model, the Resource Description Framework (RDF). RDF describes entities and attributes (edges) in the data graph, represented as RDF statements. Each statement is a triple of the form <*Subject* - *Predicate* - *Object*> [80]. The *Subject* and *Predicate* denote entities in the graph. An *Object* is either a URI or a string. Each *Predicate* URI denotes a directed attribute with *Subject* as a source and *Object* as a target.

Definition 1 [Data graph]. Formally, a data graph is a labeled directed graph $DG = \langle V, E, T \rangle$, depicting a set of RDF triples where:

- $V = \{v_1, v_2, ..., v_n\}$ is a finite set of entities;
- $E = \{e_1, e_2, ..., e_m\}$ is a finite set of edge labels;
- $T = \{t_1, t_2, ..., t_k\}$ is a finite set of triples where each triple is a proposition in the form of $\langle v_u, e_i, v_o \rangle$ with $v_u, v_o \in V$, where v_u is the *Subject* (source entity) and v_o is the *Object* (target entity); and $e_i \in E$ is the *Predicate* (edge label).

In our analysis of data graphs, the set of entities *V* will mainly consist of the concepts of the ontology and can also include individual objects (notable instances of certain concepts). The edge labels will correspond to semantic relationships between concepts and individual objects. These labels will always include the subsumption relationship `rdfs:subclassOf` and may also include the `rdf:type` relationship. For a given entity v_i, we will be interested primarily in its direct and inferred subclasses, and instances. The set of entities *V* can be divided further by using the `rdfs:subclassOf` subsumption relationship denoted as $\subseteq$) and following its transitivity inference. This includes:

- *Root entity* (r) which is superclass for all entities in the domain;
- *Category entities* ($C \subseteq V$) which is the set of all inner entities (other than the root entity r), that have at least one subclass, and may also include some individual objects (notable instances of certain concepts);
- *Leaf entities* ($L \subseteq V$) which is the set of entities that have no subclasses, and may have one or more individuals.

Starting from the root entity r, the class hierarchy in a data graph is the set of all entities linked via the subsumption relationship `rdfs:subClassOf`. The set of entities in the class hierarchy include the *root* entity r, *Category* entities C and *Leaf* entities L.

The set of edge labels E is divided further considering two relationship categories:

- *Hierarchical relationships* (H) is a set of subsumption relationships between the *Subject* and *Object* entities in the corresponding triples.
- *Domain-specific relationships* (D) represent relevant links in the domain, other than hierarchical links, e.g. in a music domain, instruments used in the same *performance* are related.

Definition 2 [Data Graph Trajectory]. A trajectory J in a data graph $DG=\langle V,E,T\rangle$ is defined as a sequence of entities and edge labels within the data graph in the form of $J=\langle v_1,e_1,v_2...,v_n,e_n,v_{n+1}\rangle$, where:

- $v_i \in V, i=1,...,n+1$;
- $e_j \in E, j=1,...,n$;
- v_1 and v_{n+1} are the first and the last entities of the data graph trajectory J, respectively;
- n is the length of the data graph trajectory J.

Definition 3 [Entity Depth]. The depth of an entity $v \in C \bigcup L$ is the length of the shortest data graph trajectory from the entity v to the root entity r in the class hierarchy of the data graph.

Definition 4 [Exploration Path]. An exploration path *P* in a data graph DG is a sequence of finite set of transition narratives generated in the form of:

$P=\left\langle\langle v_1,n_1,v_2\rangle,\langle v_2,n_2,v_3\rangle,...,\langle v_m,n_m,v_{m+1}\rangle\right\rangle$, where:

- $v_i \in V, i=1,...,m+1$;
- v_1 and v_{m+1} are the first and last entities of the exploration path P, respectively;
- m is the length of the exploration path P;
- $n_i, i=1,...,m$ is a *text string* that represents a narrative script;
- $\langle v_i,n_i,v_{i+1}\rangle$ presents a transition from v_i to v_{i+1}, which is enabled by the narrative script n_i. Note that an exploration path P is different from a data graph trajectory J in that v_i and v_{i+1} in P may not be directly linked via an edge label, i.e. the transition from v_i to v_{i+1} in P can be either via direct link, an edge, or through an implicit link, a trajectory.

Our ultimate goal is to provide an automatic way, starting from an entity $v \in V$, referred as first entity of an exploration path, and a data graph $DG=\langle V,E,T\rangle$ to generate an exploration path P in DG. For this, we will first identify entities that can serve as knowledge anchors (Section 5) and will then utilise the knowledge anchors and the subsumption strategy to generate an exploration path (Section 7).

5. Identifying Knowledge Anchors in Data Graphs

5.1. Basic Level Objects in Cognitive Science

The notion of Basic Level Objects (BLO) was introduced in Cognitive Science research, illustrating that domains of concrete objects include familiar categories that exist at an inclusive level of abstraction in humans' cognitive structures (called the *basic* level), more than categories at the *superordinate* level (i.e. above the *basic level*) or categories at the *subordinate* level (i.e. below the *basic level*) [22,62], where a human cognitive structure is hierarchically organised with respect to levels of abstraction, generality, and inclusiveness of concepts [77]. Rosch, et al .[22], define BLO as: categories that "carry the most information, possess the highest category cue validity, and are, thus, the most differentiated from one another". Crucial for identifying basic level categories is calculating *cue validity*: "the validity of a given cue *x* as a predictor of a given category *y* (the conditional probability of *y/x*) increases as the frequency with which cue *x* is associated with category *y* increases and decreases as the frequency with which cue *x* is associated with categories other than *y* increases" [22].

Most people are likely to recognise and identify objects at the basic level. An example from the experimental studies from Rosch et al. [22] of a BLO in the musical instrument domain is `Guitar`. The BLO `Guitar` represents a familiar category that is neither too generic (e.g. `musical instrument` – the root entity of the musical instrument taxonomy) nor too specific (e.g. `Folk Guitar` – subclass of the category `Guitar`). According to Rosch et al. [22], `Guitar` is at a level within the musical instrument taxonomy where its members (e.g. `Folk Guitar`, `Classical Guitar`) share attributes that are different from the attributes of members (e.g. `Grand Piano`, `Upright Piano`) of another object at the same level of abstraction (i.e. at the basic level) such

as `Piano`. This indicates that members of a BLO share many features (attributes) together, and hence they have high similarity values in terms of the feature the BLO members share. Based on the above discussion, two approaches can be applied to identify BLO in a domain taxonomy. Consequently, to identify basic level categories in a domain taxonomy, we will adopt two approaches:

Distinctiveness (highest cue validity). Identifies most differentiated category objects. A differentiated category object has most (or all) of its cues (i.e. attributes) linked to the category members (i.e. subclasses) only, and not linked to other category objects in the taxonomy. Each entity that is linked through an attribute to members of the category will have a single validity value used as a predictor of the distinctiveness of the category among other category objects in the taxonomy. The aggregation of all validity values will indicate the distinctiveness of the category object.

Homogeneity (highest commonality between category members). Identifies category objects whose members have high similarity values. The higher the similarity between category members, the more likely it is that the category object is at the basic level of abstraction. This is complementary with the distinctiveness feature. A category object with high cue validity will usually have high number of entities common to its members.

5.2. Algorithms for identifying Knowledge Anchors in Data Graphs

Our work in [81] has formally adopted the Cognitive science notion of BLO [22], to describe two groups of metrics with the corresponding algorithms for implementation. The set of all knowledge anchors in a data graph DG is denoted as KA_{DG}. The definitions of the KA_{DG} metrics followed Formal Concept Analysis (FCA) approaches in [69] and adapt them in the context of identifying KA_{DG}.

Formal Concept Analysis. Formal Concept Analysis (FCA) was presented by Wille in 1982 as a method for data analysis and knowledge representation [82]. It analyses object-attribute data tables, where data is represented as a table describing objects, attributes and binary relationships between the objects and the properties [54]. The two main notions in FCA relevant to the metrics for identifying KA_{DG} presented in this Section are: *formal context* and *formal concept*. A formal context X is represented by a triple $\langle M, R, I\rangle$, where M is a set of objects, R is a set of attributes and I is a binary relation $I \subseteq M \times R$. For an object $m \in M$ and formal attribute $r \in R$, is read as: the object m has the attribute r. Table 2 presents an example of a formal context.

Table 2. An example of a formal context in FCA represented as sets of objects M and attributes R, where (x) indicates that an object m has attribute r.

		set of *Attributes R*				
		r_1	r_2	r_3	r_4	r_5
set of *Objects M*	m_1	x		x		x
	m_2	x	x	x	x	x
	m_3		x			

For a given formal context X, let $A \subseteq R$ and $B \subseteq M$. The pair $\langle A, B\rangle$ is called a formal concept, where $A = B'$ (B' is the set of objects having all the attributes belonging to A), and $B = A'$ (A' is the set of attributes applying to all objects belonging to B). An example from the formal context X in Table 2 for a formal concept is $\langle A, B\rangle = \langle \{m_1, m_2\}, \{r_1, r_2, r_5\}\rangle$ - the concept objects m_1, m_2 are conceptually clustered based on the three shared attributes r_1, r_2, r_5.

In our work, we adapt the FCA notions of *formal context* and *formal concept* to *data graph DG* and *category entity* $v \in C$, respectively. Objects M and attributes R of a formal context $\langle M, R, I\rangle$ comprise entities V and edge labels E in a data graph DG, respectively. The objects $B \subseteq M$ of a formal concept $\langle A, B\rangle$ comprise members $v' : v' \subseteq v$ of a category entity $v \in C$, and the attributes A in the formal concept represent attribute entities v'_e linked to members $v' : v' \subseteq v$ of the category entity $v \in C$ via an edge label $e \in E$ in a data graph DG.

5.2.1. Distinctiveness Metrics

This group of metrics aims to identify the most differentiated category entities whose members are linked to distinctive entities that are shared amongst the members of the category entities but are not shared to members of other categories. Each entity $v \in V$ that is linked through an edge label e to members $v' \subseteq v$ of the category entity $v \in C$ will

have a single validity value used to distinctive the category entity $v \in C$ among other category entities in the data graph). Three distinctiveness metrics were developed:

Attribute Validity (AV). The attribute validity definition corresponds to the cue validity definition in [22] and adapts the formula from [69]. We use 'attribute validity' to indicate the association with data graphs - 'cues' in data graphs are attributes of the entities and are represented as relationships in terms of triples. The attribute validity value of an entity $v \in C$ is calculated with regard to a relationship type *e*, as the aggregation of the attribute validity values for all entities v'_e linked to subclasses $v' : v' \subseteq v$. The attribute validity value of v'_e *increases*, as the number of relationships of type *e* between v'_e and the *subclasses* $v' : v' \subseteq v$ increases; whereas the attribute validity value of v'_e *decreases* as the number of relationships of type *e* between v'_e and *all entities* in the data graph increases.

We define the set of vertices *W(v,e)* related as *Subjects* to the *subclasses* $v' : v' \subseteq v$, via relationship of type *e* :

$$W(v,e) = \{\ v'_e :\ \exists v' \left[v' \subseteq v \wedge \langle v'_e, e, v' \rangle \in T \right] \} \qquad (1)$$

The following formula defines the attribute validity metric for a given entity *v* with regard to a relationship type *e*.

$$AV(v,e) = \sum_{v'_e \in W(v,e)} \frac{|\{\langle v'_e, e, v' \rangle : v' \subseteq v\}|}{|\{\langle v'_e, e, v_a \rangle : v_a \in V\}|} \qquad (2)$$

In Figure 3, the *AV* value for category entity v_2 with regard the domain-specific relationship D is the aggregation of the *AV* values of the (*Subject* entities e_3, e_4, e_5, e_6) linked to members of the category entity v_2 (*Object* entities $v_{21}, v_{22}, v_{23}, v_{24}$) via the edge label (i.e. *Predicate*) D. The *AV* value for the entity e_3 equals the number of the triples between the *Subject* entity e_3 and members of the category v_2 (*Object* entities v_{21}, v_{22}) via the relationship D (2 triples), divided by the number of triples between the *Subject* entity e_3 and all *Object* entities in the graph (*Object* entities v_{12}, v_{21}, v_{22}) via the relationship D (3 triples). Hence the *AV* value for e_3 equals 2/3 = 0.66.

The aggregation *AV* values for entities e_3, e_4, e_5, e_6 will identify the *AV* value for the category entity v_2

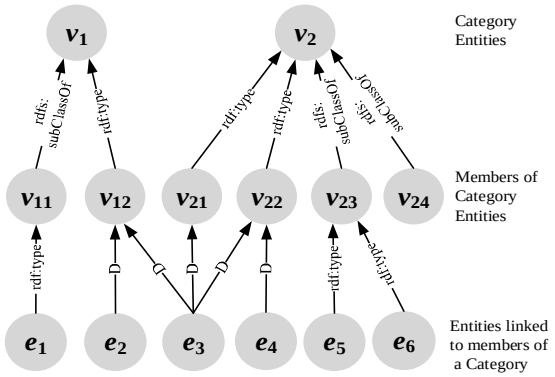

Fig. 3. A data graph showing entities and relationship types between entities.

Category Attribute Collocation (CAC). This approach was used in [83] to improve the cue validity metric by adding the so called category-feature collocation measure which takes into account the frequency of the attribute within the members of the category. This gives preference to 'good' categories that have many attributes shared by their members. In our case, a good category will be an entity $v \in C$ with high number of relationships of type *e* between v'_e and the subclasses $v' : v' \subseteq v$, relative to the number of its subclasses. The following formula defines the category-attribute collocation metric for a given entity v with regard to a relationship type e.

$$CAC(v,e) = \sum_{v'_e \in W(v,e)} \frac{|\{\langle v'_e, e, v' \rangle : v' \subseteq v\}|}{|\{\langle v'_e, e, v_a \rangle : v_a \in V\}|} \cdot \frac{|\{\langle v'_e, e, v' \rangle : v' \subseteq v\}|}{|V'|} \qquad (3)$$

Considering the above example for identifying the *AV* value for the entity e_3, and considering the relationship D, the *CAC* adds a weight of (the number of the triples the *Subject* between e_3 and the members of v_2 (*Object* entities v_{21}, v_{22}) via the relationship D (2 triples) divided by (the number of members of the category entity v_2 (i.e. FOUR members: $v_{21}, v_{22}, v_{23}, v_{24}$). Hence the *CAC* of entity e_3 will be the *AV* value of e_3 multiplied by (2/4). The *CAC* value for entity e_3 equals [(2/3) * (2/4) = 0.33]. The aggregation *CAC* values for entities e_3, e_4, e_5, e_6 will identify the *CAC* value for the category entity v_2.

Category Utility (CU). This approach was presented in [84] as an alternative metric for obtaining categories at the basic level object. The metric takes into account that a category is useful if it can improve the ability to predict the attributes for members of the category, i.e. a good category will have many attributes shared by its members (as mentioned in the category-attribute collocation metric). At the same time, it possess 'unique' attributes that are not related to many other categories (efficiency of category recognition). We adapt the formula in [69] for a data graph:

$$CU(v,e)=\frac{|V'|}{|V|}\sum_{v'_e \in V(v,e)}\left(\frac{|\{\langle v'_e,e,v\rangle : v\subseteq v\}|}{|V'|}\right)^2-\left(\frac{|\{\langle v'_e,e,v_a\rangle : v_a \in V\}|}{|V|}\right)^2 \quad (4)$$

Following the previous examples for calculating the AV and the CAC values for entity e_3 (see Figure 3); in addition to the proportion used by the CAC value (2/4), the CU will include the proportion of all triples between the *Subject* entity e_3 and all *Object* entities in the graph (THREE *Object* entities v_{12}, v_{21}, v_{22}) linked via relationship D (i.e. 3 triples) over the number of entities linked via subsumption relationships (e.g. `rdfs:subClassOf` and `rdf:type`) in the graph (11 entities). Hence the CU value for e_3 will be: $[(2/4)^2 - (3/11)^2 = 0.177]$.

The aggregation CU values for entities e_3, e_4, e_5, e_6, multiplied by the number of members of category v_2 (FOUR members) divided by number of entities in the graph linked via the susbsumption relationships (11 entities) will identify the CU value for the category entity v_2.

Algorithm I describes calculating the distinctiveness metrics. The algorithm takes a data graph *DG* and a relationship type (hierarchical or domain-specific relationship) as input and returns values for the three distinctiveness metrics for each category entity $v \in C$.

In (line 1), all category entities are retrieved via SPARQL query:

```
SELECT distinct ?category
WHERE {
 ?category rdfs:subClassOf r.
 ?subclass rdfs:subClassOf ?category.}
```

where *r* is the root entity in the data graph.

After tha, for an entity *v*, all members are retrieved (line 2). Members of an entity can be subclasses or instances. In the current implementation, we are using the subsumption relationship `rdfs:subClassOf` to retrieve the entity's members via the following SPARQL query:

```
SELECT distinct ?subclass
WHERE {
 ?subclass rdfs:subClassOf  v.}
```

For each entity v'_e linked to one or more subclass entities v' via an edge label e $\langle v'_e, e, v'\rangle$ (line 3), several steps are conducted: retrieving all triples with *Subject* v'_e and *Object* any subclass v' (line 4); retrieving all triples with *Subject* v'_e and *Object* any graph entity *v* (line 5); applying the formulas for calculating the *AV*, *CAC*, and *CU* metrics for v'_e (lines 6-8); and aggregating values for v'_e to the overall values for *v* (lines 9-11).

Algorithm I: Distinctiveness Metrics

Input: $DG = \langle V, E, T\rangle, e \in E$

Output: AV_v, CAC_v, CU_v for all $v \in C$

1. **for all** $v \in C$ **do** *//all category entities*
2. V' := the set of all $v' : v' \subseteq v$ *//all members of v*
3. **for all** $v'_e : \exists\langle v'_e, e, v'\rangle$ **do**
4. N_e := set of all $\langle v'_e, e, v'\rangle : v' \in V'$
5. M_e := set of all $\langle v'_e, e, v_a\rangle : v_a \in V$
6. $AV_{v'_e} := |N_e| / |M_e|$
7. $CAC_{v'_e} := (|N_e| / |M_e|) \cdot (|N_e| / |V'|)$
8. $CU_{v'_e} := (|N_e| / |V'|)^2 - (|M_e| / |V|)^2$
9. $AV_v := AV_v + AV_{v'_e}$
10. $CAC_v := CAC_v + CAC_{v'_e}$
11. $CU_v := CU_v + CU_{v'_e}$
12. **end for**
13. $CU_v := \frac{|V'|}{|V|} \cdot CU_v$
14. **end for**

5.2.2. Homogeneity Metrics

These metrics aim to identify categories whose members share many entities among each other. In this work, we have utilised three set-based similarity metrics [81]: *Common Neighbors (CN), Jaccard (Jac), and Cosine (Cos).* The homogeneity value for a category entity is identified as the average of the accumulated pair-wise similarity values between the categories members with regard to an edge label e. For instance (see Figure 3), the *Jaccard* similarity between the pair-wise members (v_{21}, v_{22}) of the entity v_2 considering the edge label D is the number of intersected *Subject* entities (ONE entity e_3) linked to

Objects v_{21}, v_{22} via edge label D, divided by the number of union *Subject* entities (TWO entities e_3, e_4) linked to *Objects* v_{21}, v_{22} via edge label D. Accordingly, the *Jaccard* similarity between members (v_{21}, v_{22}) is (1/2).

Algorithm II describes implementation of the metrics. The algorithm takes a data graph and a relationship type (hierarchical or domain-specific relationship) as input and returns values for the three homogeneity metrics for each entity $v \in C$

Algorithm II: Homogeneity Metrics

Input: $DG = \langle V, E, T \rangle, e \in E$

Output: CN_v, Jac_v, Cos_v for all $v \in C$

1. **for all** $v \in C$ **do** *//all category entities*
2. V' := the set of all $v' : v' \subseteq v$ *//all members of v*
3. **for all** $(v', v'') : v' \in V' \wedge v'' \in V'$ **do**
4. $V_e' := \{v_e' : \exists \langle v_e', e, v' \rangle\}$
5. $V_e'' := \{v_e'' : \exists \langle v_e'', e, v'' \rangle\}$
6. $I := V_e' \cap V_e''$
7. $U := V_e' \cup V_e''$
8. $CN_{v',v''} := |I|$ *//Common Neighbors*
9. $Jac_{v',v''} := |I| / |U|$ *//Jaccard*
10. $Cos_{v',v''} := |I| / (\sqrt{|V_e'|} \cdot \sqrt{|V_e''|})$ *//Cosine*
11. $CN_v = CN_v + CN_{v',v''}$
12. $Jac_v = Jac_v + Jac_{v',v''}$
13. $Cos_v = Cos_v + Cos_{v',v''}$
14. **end for**
15. $CN_v = CN_v / (|V'| . (|V'| - 1) / 2)$
16. $Jac_v = Jac_v / (|V'| . (|V'| - 1) / 2)$
17. $Cos_v = Cos_v / (|V'| . (|V'| - 1) / 2)$
18. **end for**

The algorithm takes a data graph and a relationship type (hierarchical or domain-specific relationship) as input and returns values for the three homogeneity metrics for each entity $v \in C$. In (line 1), all category entities are retrieved via the same SPARQL query used in (Algorithm I / line 1).

Then, for every entity *v*, all members are retrieved using the subsumption relationship (line 2) using the same SPARQL query in (Algorithm I / line 2). For each pair of subclass entities v' and v'' (line 3), several steps are conducted: retrieving all entities linked via triples with v' and v'' (lines 4-5); calculating their intersection and union (lines 6-7); applying the formulas for calculating the similarity metrics *CN*, *Jac*, and *Cos* (lines 8-10); and aggregating these values to the overall values for *v* (lines 11-13); and normalising the aggregated values (lines 15-17).

All SPARQL queries were run over the MusicPinta data graph [18] stored in a triple store. This implementation allowed examining the performance of the KA_{DG} metrics over a specific data graph, as presented next.

6. Evaluating KA_{DG} against Human BLO_{DG}

To enable impartial comparison of the outputs by the KA_{DG} algorithms and the cognitive structures of humans, we conducted a study with humans following earlier Cognitive Science studies closely to identify basic level objects in domain taxonomies. As a use case in a representative domain for evaluating knowledge anchors over a data graph, we used a typical semantic data browser, MusicPinta, which was developed in our earlier research [18] and is introduced in section 3. We focus on musical instruments since they are objects rich in meaning and provide a suitable domain for studying BLO, as discussed in [85]. It should be noted that it is not appropriate to take the list of BLO derived empirically by earlier Cognitive Science studies (e.g. [22,85]) in music domain because these BLO presents a *commonsense* view of the domain (music in this case) that may not correspond to the data graph over which the algorithms are run (data graphs are *partial abstract models* of the real world). Hence, to ensure that the evaluation focuses on the performance of the algorithms, we have to eliminate any impact of the knowledge modelling in the data graph (e.g. missing or wrong concepts/relationships).

6.1. Cognitive Science Experimental Approaches for Deriving BLO

While studying the notion of basic level objects, Rosch et al [22] conducted several experiments comprising free-naming tasks testing the hypothesis that object names at the basic level should be the names by which objects are most generally designated by adults. In a free-naming task, objects in a domain taxonomy are shown to a participant as a series of images in a fixed amount of time, and the participant is asked to identify the names of objects shown in the images as quickly as possible. Three types of packets of images were shown to the participants in their experiments: those in which one picture from each superordinate category appeared; one in which one im-

age from each basic level category appeared; and one in which all images appeared. The participants overwhelmingly used names at the basic level while naming objects in the images [22].

To identify BLO, they considered accuracy and frequency. *Accuracy* refers to naming an entity correctly by the user. It considers whether a user names an entity with its *exact name*, or with a *parent* (superclass) or with a category *member* (subclass) of the entity. *Frequency* indicates *how many times a particular category was accurately named by different users*. In the example of `Guitar`, when participants were shown members of `Guitar` object (e.g. `Folk Guitar`, `Classical Guitar`) in a packet, they named them with their parent `Guitar` at the basic level more frequently than with names at the superordinate level (e.g. `Musical instrument`) or with their exact names (e.g. `Folk Guitar`, `Classical Guitar`) at the subordinate level.

The selection of object names used in the free-naming tasks in [22] was based on the population of categories of concrete nouns in common use in English. Every noun with a word frequency of ten or greater from a sample of written English [86] was selected as a basic level object. A superordinate category was considered in common use if at least four of its members met this criterion.

However, the Cognitive Science approach for selecting BLO cannot be applied directly in the context of a data graph. The principal difference is that we need to constrain the human cognitive structures upon the data graph, as opposed to using a bag of words from popular dictionaries. This is because a data graph presents a lesser number of concepts from a domain, which belong to the graph scope, and there can be concepts that have been omitted. Moreover, the Cognitive Science studies included concrete domains where images of the objects could be shown to participants. Many semantic web applications utilise data graphs which include more abstract concepts for which images cannot be reliably shown to users (e.g. medical illnesses, environmental concepts, professions). Therefore, we adapt the Cognitive Science experimental approach for deriving BLO to consider the domain coverage of a data graph, which is applicable to any domain presented with a data graph.

6.2. Algorithm for Identifying Human Basic Level Objects in a Data Graph

Following Cognitive Science experimental studies outlined above, we present two strategies with the corresponding algorithm for identifying human basic level objects in a data graph (BLO_{DG}).

Strategy 1. Takes into account whether a leaf entity $v \in L$ that has no subclasses is presented to a user and named with one of its parents (i.e. super classes).

Strategy 2. Takes into account whether a category entity $v \in C$ that has one or more subclasses is presented and named with its exact name, or with the name of a category parent (i.e. superclass) or a category member (i.e. subclass that is not a leaf entity).

Algorithm III describes the two strategies for identifying human BLO_{DG} using *accuracy* and *frequency*. *Accuracy* refers to naming an entity correctly.

Algorithm III: Identifying Human BLO_{DG}

Input: $DG = \langle V, E, T \rangle$

Output: two sets of entities: ***Set1*** for human BLO_{DG} identified from *Strategy 1*, and ***Set2*** for **human** BLO_{DG} identified from *Strategy 2*. The **Union** of the ***Set1*** *and* ***Set2*** identifies the final set of Human BLO_{DG}

1. **for a set of entities** $v \subseteq V$ **do**
2. **for all** $(i := 1; i \leq n; i++)$
3. **if** $v \in L$ **then** ***//Strategy1***
4. $show(v)$ **and ask a user to name** v
5. **if** $answer(v) \in parent(v)$ **then**
6. $count_a++;$ *//count frequency*
7. **end if;**
8. **else if** $v \in C$ **then** ***//Strategy2***
9. $show(v)$ **and ask a user to name** v
10. **if** $answer(v) = label(v)$ **then**
11. $count_a++;$ *//count frequency*
12. **else if** $answer(v) \in \{parent(v) \cup member(v)\}$ **then**
13. $count_a++;$ *//count frequency*
14. **end if;**
15. **end if;**
16. **end for;**
17. **end for;**
18. ***Set1*** = $\{answer(v) : v \in L \wedge count_a \geq k\}$
19. ***Set2*** = $\{answer(v) : v \in C \wedge count_a \geq k\}$

The algorithm takes a data graph *DG* as input and returns two sets of human BLO_{DG}. For any class entity $v \subseteq V$ (line 1), we identify the number of users to be asked to name the entity (line 2). For *Strategy 1* (lines 3-7), we consider accurate naming of a category entity (a parent) when a leaf entity $v \in L$ that is a member of this category is seen. For *Strategy 2* (lines 8-14), we consider naming a category entity $v \in C$ with its exact name (lines 10, 11) or a name of its parents (superclasses) or category members (subclasses that are not leaf entities) (lines 12-13).

In both the strategies, we use a representation function $show(v)$ to create a representation of an entity v to be shown to the user. The representation of a leaf entity $v \in L$ (in *Strategy* 1) will consider the leaf itself (e.g. show a single label or a single image for the leaf entity), while the representation of a category entity $v \in C$ (in *Strategy* 2) will consider all (or some) of the category's leaf entities (e.g. showing a random listing of a set of labels of leaf entities or showing a group of images of leaf entities as a collage). The following SPARQL query is used to get the set of leaf entities of v :

```
SELECT ?leaf ?leaf_label
 WHERE {
   ?leaf rdfs:subClassOf v .
   ?leaf rdfs:label ?leaf_label.
 FILTER NOT EXISTS
    {?member rdfs:subClassOf ?leaf.}}
```

The two strategies in Algorithm III for obtaining human BLO_{DG} are applied as follows:

Strategy 1. When a user is shown a representation of a leaf entity $v \in L$ (line 4), the following steps are conducted:

- The function $answer(v)$ assigns a user's *answer* to the leaf entity v .
- The function $parent(v)$ returns a set of labels (i.e. names) of the parent(s) of the leaf entity v via the following SPARQL query:

```
SELECT ?parent_label ?label
 WHERE {
    v rdfs:subClassOf ?parent.
   ?parent rdfs:label ?parent_label.}
```

- The algorithm III (line 5) checks if the user named the leaf entity v with one of its parents. If an accurate name of a parent was provided, then the frequency of the parent entity will be increased by one (line 6).

Strategy 2. When a user is shown a representation of a category entity $v \in C$ (line 9), the following steps are conducted:

- The function $answer(v)$ assigns a user's *answer* to the category entity v .
- The function $parent(v)$ returns a set of labels of *parent*(s) of the category entity v via SPARQL queries similar to *Strategy* 1 above.
- The function $member(v)$ returns a set of labels (i.e. names) of member(s) of the category entity v via the following SPARQL query:

```
SELECT ?member_label
 WHERE {
  ?member rdfs:subClassOf v .
  ?member rdfs:label ?member_label.}
```

- The function *label*(v) returns the *label* of the category entity v via the following SPARQL query:

```
SELECT ?label
 WHERE {
   v rdfs:label ?label.}
```

The algorithm in (lines 10, 12) checks if the user named the category entity v with its exact name, or a name of its parents or its members. If there was accurate naming of the category, a parent or a member, the frequency of the category name (line 11), the parent name or the member name (line 13) will be increased by one. *K* in lines 18 & 19 indicate the number of different participants that have correctly named a category as indicated in lines (11,13).

6.3. Evaluating KA_{DG} *against human* BLO_{DG}

6.3.1. Obtaining human BLO_{DG}

To obtain a set of human BLO_{DG} that correspond to a human cognitive structure, we conducted a user study in the musical instrument domain to apply Algorithm III.

Participants. For this user study, 40 participants – university students and professionals, age 18–55, were recruited on a voluntary basis. None of the participants had any expertise in music.

Method. The participants were asked to freely name objects that were shown in image stimuli, under limited response time (10 seconds). Overall, 364 taxonomical musical instruments were extracted from the MusicPinta dataset by running SPARQL queries over the triple-store hosting the dataset to get all musical instrument concepts linked via the `rdfs:subclassOf` relationship. The entities included: *leaf entities* (total 256) and *category entities* (total 108). While applying the two strategies in Algorithm III, for each leaf entity, we collected a representative image from the Musical Instrument Museums Online (MIMO)[8] to ensure that pictures of high quality were shown[9]. For a category entity, all leaves from that category entity were shown as a group in a single image (similarly to a packet of images in [22]).

[8] http://www.mimo-international.com/MIMO/

[9] MIMO provided pictures for most musical instruments. In the rare occasions when an image did not exist in MIMO, Wikipedia images were used instead.

We ran ten online surveys[10]. Among them: (i) eight surveys presented 256 leaf entities, each showed 32 leaves; (ii) two surveys presented 108 category entities, each showed 54 categories.

Free-naming task. Each image was shown for 10 seconds on the participant's screen, and the participant was asked to type the name of the given object (for leaf entities) or the category of objects (for category entities). The image allocation in the surveys was random. Every survey had four respondents from the study participants (corresponds to line 2 in Algorithm III). Each participant was allocated only to one survey (either leaf entities or category entities). Figures 4-7 show example instrument images and participant answers (Figure 4 from Strategy 1, and Figures 5-7 from Strategy 2). Applying Algorithm III over the MusicPinta dataset, two sets of human BLO_{DG} were identified.

Set1 [resulting from Strategy1]. We consider accurate naming of a category entity (parent) when leaf entity that belongs to this category is seen. For example, as shown in Figure 4, a participant has named `Violotta`, a leaf entity in the data graph, with its parent category `Violin`. This will be counted as an accurate naming and will increase the count for `Violin`. The overall count for `Violin` will include all cases when participants named `Violin` while seeing any of its leaf members.

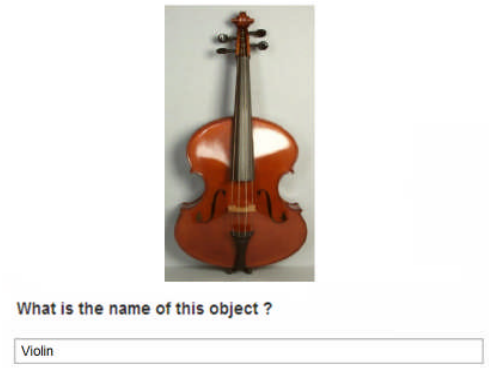

Fig. 4. An image of `Violotta` (a leaf entity in the data graph) was shown to a user, who named it as `Violin`.

Set2 [resulting from Strategy 2]. We consider naming a category entity with its exact name or a name of its parent or subclass member. For example (see Figure 5), a participant saw the category `Fiddle` and named its parent category `Violin`; this will increase the count for `Violin`. In Figure 6 a participant was shown the image of category `Violin` and named it with its exact name; this will increase the count for `Violin`.

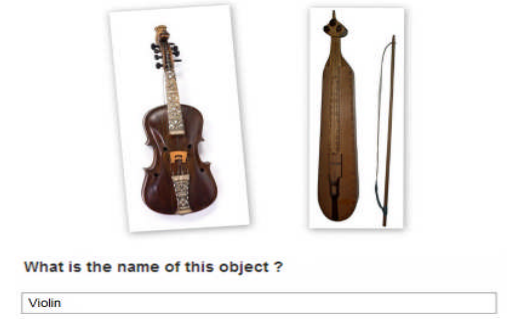

Fig. 5. An image of `Fiddle` (a category entity with two leaf entities) was shown to a user, who named it as `Violin`.

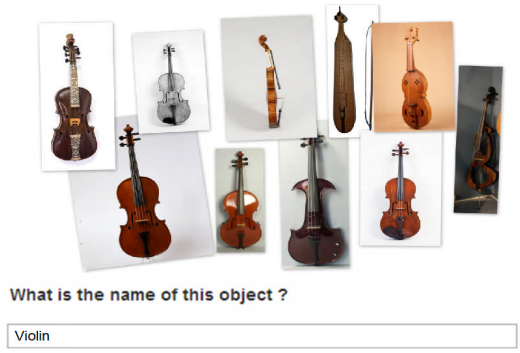

Fig. 6. An image of `Violin` (a category entity) was shown to a user, who named it as `Violin`

In Figure 7 a participant saw the category `Bowed String Instrument` and named it as its member category `Violin`; this will also increase the count for `Violin`.

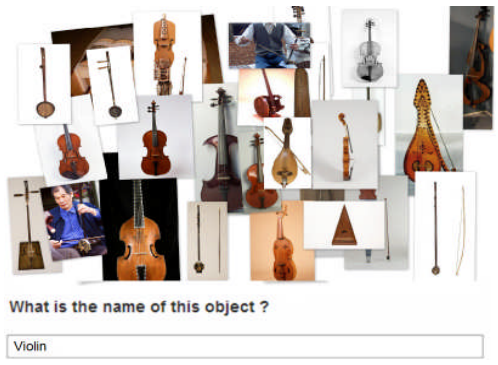

Fig. 7. An image of `Bowed String Instrument` (a category entity) was shown to a user, who named it as `Violin`

[10] The study was conducted with Qualtrics (www.qualtrics.com). Examples from the surveys are in: https://login.qualtrics.com/jfe/preview/SV_cHhHPPthBFO5r6d?Q_CHL=preview

In each of the two sets, entities with frequency equal or above two (i.e. named by at least two different users) were identified as potential human BLO_{DG}. The union of *Set1* and *Set2* gives human BLO_{DG} (we identified 24 human BLO_{DG}). The full list of human BLO_{DG} obtained from MusicPinta is available in [87].

6.3.2. Quantitative Analysis

We used the identified human BLO_{DG} to examine the performance of the KA_{DG} metrics. For each metric, we aggregated (using union) the KA_{DG} entities identified using the hierarchical relationships (*H*). We noticed that the three homogeneity metrics have the same values; therefore, we chose one metric when reporting the results, namely Jaccard similarity[11]. A cut-off threshold point for the result lists with potential KA_{DG} entities was identified by normalizing the output values from each metric and taking the mean value for the *60th percentile* of the normalised lists[12]. The KA_{DG} metrics evaluated included the three distinctiveness metrics plus the Jaccard homogeneity metric; each metric was applied over both families of relationships – hierarchical (*H*) and domain-specific (*D*).

As in ontology summarisation approaches [60], a name simplicity strategy based on data graphs was applied to reduce noise when calculating key concepts (usually, basic level objects have relatively simple labels, such as chair or dog).

The name simplicity approach we use is solely based on the data graph. We identify the *weighted median* for the length of the labels of all data graph entities $v \subseteq V$ and filter out all entities whose label length is higher than the median. For the MusicPinta data graph, the weighted median is 1.2, and hence we only included entities which consist of one word. Table 3 illustrates precision and recall values comparing human BLO_{DG} and KA_{DG} derived using hierarchical and domain specific relationships.

Table 3. MusicPinta: performance of the KA_{DG} algorithms compared to human BLO_{DG}.

Relationship types	Precision				Recall			
	AV	CAC	CU	Jac	AV	CAC	CU	Jac
H	0.60	0.62	0.62	0.60	0.68	0.73	0.73	0.55
D	0.55	0.53	0.55	0.62	0.50	0.45	0.50	0.36

[11] The Jaccard similarity metric is widely used, and was used in identifying basic formal concepts in the context of formal concept analysis [68].

[12] We experimented the KA_{DG} metrics at the 60^{th}, 70^{th}, 80^{th} and 90^{th} percentiles on the metrics normalised output lists, and the metrics performed best using at the 60^{th} percentile.

Hybridisation of Algorithms. Further analysis of the False Positive(FP) and False Negative (FN) entities indicated that the algorithms had different performance on the different taxonomical levels in the data graph. This led to the following heuristics for hybridisation of algorithms.

Heuristic 1: Use Jaccard metric with hierarchical relationships for the most specific categories in the graph (i.e. the categories at the bottom quartile of the taxonomical level). There were FP entities (e.g. `Shawm` and `Oboe`) returned by distinctiveness metrics using the domain-specific relationship `MusicOntology:Performance` because these entities are highly associated with musical performances (e.g. `Shawm` is linked to 99 performances and `Oboe` is linked to 27 performance). Such entities may not be good knowledge anchors for exploration, as their hierarchical structure is flat. The best performing metric at the specific level was Jaccard for hierarchical attributes - it excluded entities which had no (or a very small number of) hierarchical attributes.

Heuristic 2: Take the majority voting for all the other taxonomical levels. Most of the entities at the middle and top taxonomical level will be well represented in the graph hierarchy and may include domain-specific relationships. Hence, combining the values of all algorithms is sensible. Each algorithm represents a voter and provides two lists of votes, each list corresponding to hierarchical or domain-specific associated attributes (*H*, *D*). At least half of the voters should vote for an entity for it to be identified in KA_{DG}. Examples from the list of KA_{DG} identified by applying the above hybridisation heuristics included `Accordion`, `Guitar` and `Xylophone`. The full KA_{DG} list is available here [87]. Applying the hybridisation heuristics improved *Precision value* to **0.65** (average Precision in Table 3 = 0.59), *Recall value* to **0.68** (average Recall in Table 3 = 0.56).

6.3.3. Qualitative Analysis

Examining the FP and FN entities for the hybridization algorithm, we made the following observations that relate to the possible use of KA_{DG} as exploration anchors.

Observation 1: Missing basic level entities due to unpopulated areas in the data graph. We noticed that none of the metrics picked FN entities (such as `Harmonica, Banjo` or `Cello`) that belonged to the bottom quartile of the class hierarchy and had a small number of subclasses (e.g. `Harmonica, Banjo` and `Cello` each have only one subclass

and there are no domain-specific relationships with their members). Similarly, none of the metrics picked `Trombone` (which is false negative) - although `Trombone` has three subclasses, it is linked only to one performance and is not linked to any DBpedia categories. While these entities belong to the cognitive structures of humans and were therefore added in the benchmarking sets, one could *doubt* whether such entities would be useful exploration anchors because they are not sufficiently presented in the data graph. These entities would take the user to *'dead-ends'* with unpopulated areas which may be confusing for exploration. We therefore argue that such FN cases could be seen as 'good misses' of algorithms.

*Observation 2: Selecting entities that are superordinate of basic level entities***.** The FP included entities, such as `Tambura`, `Reeds`, `Bass`, `Brass`, `Castanets`, and `Woodwind`, which are well presented in the graph hierarchy (e.g. `Reeds` has 36 subclasses linked to 60 DBpedia categories, `Brass` has 26 subclasses linked to 22 DBpedia categories, `Woodwind` has 72 subclasses linked to 82 DBpedia categories). Also, their members participate in many domain-specific relationships (e.g. `Reeds` members are linked to 606 performances, `Brass` - 33, and `Woodwind` - 853). Although, these entities are not close to the human cognitive structures, they provide direct links KA_{DG} entities from the benchmarking sets (e.g. Reeds links to `Accordion`, `Brass` links to `Trumpet` and `Woodwind` links to `Flute`). We therefore argue that such FP cases could be seen as 'good picks' of the algorithms because they can provide exploration *bridges* to reach BLO.

7. Exploration Strategies Based on Subsumption

In this section we describe how we adopt the subsumption theory for meaningful learning [21] (described in section 3) as our underpinning theoretical model to develop an automatic approach for generating exploration paths in data graphs.

7.1. Challenges in Applying the Subsumption Theory for Meaningful Learning

Applying the subsumption theory for meaningful learning to generate exploration paths brings forth two challenges:

- *How to find the closest knowledge anchor to the first entity of an exploration path?* In uni-focal browsing (pivoting), a user starts his/her exploration from a single entity in the graph, also referred as a first entity (v_s). To start the subsumption process, the user has to be directed from this first entity to a suitable knowledge anchor in the data graph from where links to new entities can be made. However, there can be several knowledge anchors in a data graph, requiring a mechanism to identify the closest knowledge anchor v_{KA} to v_s.
- *How to use the closest knowledge anchor to subsume new class entities for generating an exploration path?* The closest knowledge anchor usually can have many subclass entities that exist at different levels of abstractions. It is important to identify which subclasses to subsume and in what order while generating an exploration path for the user. Furthermore, we also need to identify appropriate narrative scripts between the entities in the exploration path. We argue that providing meaningful narrative scripts between entities will help layman users to create meaningful relationships between familiar entities they already know in their cognitive structures and the new subsumed class entities.

7.2. Finding the Closest KA_{DG}

Let v_s be the first entity of an exploration path. The first entity v_s can be any class entity in the graph (i.e. any entity in the subsumption class hierarchy of all entities linked via the subsumption relationship `rdfs:subClassOf`). If v_s *is a knowledge anchor* ($v_s \in KA_{DG}$), then there is no need to identify the closest knowledge anchor (i.e. find another anchor in the data graph), and the subsumption process can start immediately from v_s. However, if v_s *is not a knowledge anchor* ($v_s \notin KA_{DG}$), then v_s can be *superordinate*, *subordinate*, or *sibling* of one or more knowledge anchors. In this case, an automatic approach for identifying the closest knowledge anchor v_{KA} to v_s is required.

To find the closest knowledge anchor, we calculate the semantic similarity between v_s and every knowledge anchor in the data graph $v_i \in KA_{DG}$. The semantic similarity between two entities in the class hierarchy is based on their distances (i.e. length of the data graph trajectory between both entities). Due to the fact that class hierarchies exist in most data graphs, we adopt the semantic similarity metric from [88] and apply it in the context of a data graph, where semantic similarity is based on the lengths between the entities in the class hierarchy. The semantic simi-

larity between v_s and a knowledge anchor v_i is calculated as:

$$sim(v_s, v_i) := \frac{2.depth(lca(v_s, v_i))}{depth(v_s) + depth(v_i)} \quad (5)$$

where, $lca(v_s, v_i)$ is the *least common ancestor* of v_s and v_i, and $depth(v)$ is a function for identifying the depth of the entity v in the class hierarchy.
Algorithm IIII describes how the semantic similarity metric is applied to identify v_{KA}.

Algorithm IIII: Identifying Closest KA$_{DG}$

Input: $DG = \langle V, E, T \rangle$, $v_s \in V$, $KA_{DG} = \{v_1, v_2, ..., v_i\}$

Output: v_{KA} – closest knowledge anchor with highest semantic similarity to v_s

1. **if** $v_s \in KA_{DG}$ **then** *// v_s is a knowledge anchor*
2. $v_{KA} := v_s$;
3. **else** *// v_s is NOT a knowledge anchor*
4. $S := \{\}$; *//list for storing similarity values*
5. **for all** $v_i \in KA_{DG}$ **do** *//for all knowledge anchors*
6. $CA := \{\}$; *//list to store common ancestors of v_s and v_i*
7. $L := \{\}$; *//list for storing trajectory lengths*
8. $CA \leftarrow common\_ancestors(v_s, v_i)$;
9. **for all** $v_{ca} \in CA$ *//for all common ancestors*
10. $L \leftarrow length(v_{ca}, v_i)$; *//length between v_{ca} and v_i*
11. **end for;**
12. $v_{lca} := v_{ca}$ with least length in L;
13. $S \leftarrow \frac{2 \cdot depth(v_{lca})}{depth(v_s) + depth(v_i)}$;
14. **end for;**
15. $v_{KA} := v_i$ with maximum similarity value in list *S*;
16. **end if;**

The algorithm takes a data graph, the first entity v_s of an exploration path and a set of knowledge anchors KA$_{DG}$ as an input, and identifies the closest knowledge anchor $v_{KA} \in$ KA$_{DG}$ with highest semantic similarity value to v_s. If the first entity v_s belongs to the set of knowledge anchors $v_s \in$ KA$_{DG}$ (line 1), then the first entity v_s is identified as the closest knowledge anchor v_{KA} (line 2). However, if the first entity v_s does not belong to the set of knowledge anchors in the data graph $v_s \notin$ KA$_{DG}$ (line 3), then the following steps are conducted:

- The algorithm initialises a list *S* to store semantic similarity values between v_s and every knowledge anchor $v_i \in$ KA$_{DG}$ (line 4).
- For every knowledge anchor $v_i \in$ KA$_{DG}$ (line 5), the algorithm initiates two lists: list *CA* for storing the common ancestors (i.e. common superclasses) of v_s and v_i (line 6), and list L for storing the trajectory lengths between the common ancestors in list *CA* and the knowledge anchor v_i (line 7).
- The algorithm in (line 8) uses a function *common_ancestors*(v_s,v_i) which retrieves all common ancestors of v_s and v_i in the class hierarchy, and stores them in list *CA*. The function uses the following SPARQL query to retrieve the common ancestors of v_s and v_i:

```
SELECT distinct ?common_ancestor
 WHERE {
   vs rdfs:subClassOf ?common_ancestor.
   vi rdfs:subClassOf ?common_ancestor}.
```

- For every common ancestor in list CA (line 9), the algorithm identifies the *length* of the data graph trajectories between v_i and each of the common ancestors v_{ca} in list CA, via the SPARQL query:

```
SELECT(count(?intermediate)-1 as ?length)
 WHERE {
   vi rdfs:subClassOf ?intermediate.
   ?intermediate rdfs:subClassOf vca.}
```

- The common ancestor v_{ca} with least trajectory length with v_i in list L is identifies as the *least common ancestor* v_{lca} (line 12).
- After identifying the least common ancestor v_{lca} to the first entity v_s and the knowledge anchor v_i, the semantic similarity metric (Formula 5) is applied (line 13). The metric includes identifying depths of v_s, v_i, and v_{lca}. The depth of an entity v is identified using the following SPARQL query:

```
SELECT (count(?intermediate)-1 as ?depth)
 WHERE {
   v rdfs:subClassOf ?intermediate.
   ?intermediate rdfs:subClassOf r .}
```

where r is the root entity in the data graph.

The semantic similarity value is then inserted into the list *S* (line 13), and the knowledge anchor with the highest similarity value to the first entity v_s will be identified as closest knowledge anchor v_{KA} (line 15).

7.3. Subsumption Using the Closest Knowledge Anchor

The closest knowledge anchor v_{KA} is used to subsume new class entities and to generate transition narratives in the exploration path. Table 4, describes the different conditions for the narrative scripts used between entities in an exploration path.

Table 4. Narrative types of transitions between entities in an exploration path.

Narrative Type	From entity	To entity	Description	Output script *(From_ entity, Narrative type, To_entity)*
N_1	v_s	v_{KA}	v_s is subclass of v_{KA}	*"You may find it useful to know that v_s belongs to a familiar and well-known class–v_{KA}. Let's explore v_{KA}."*
N_2	v_s	v_{KA}	v_s is superclass of v_{KA}	*"You may find it useful to know that there is a well-known class – v_{KA} that belongs to v_s . Let's explore v_{KA}"*
N_3	v_s	v_{KA}	v_s and v_{KA} Are siblings	*"You may find it useful to know that v_s is similar to a well-known class – v_{KA}. Let's explore v_{KA}."*
N_4	v_{KA}	v'_{KA}	v'_{KA} is subclass of v_{KA} and superclass of v_s	*"You may find it useful to know that v'_{KA} belongs to v_{KA}, and v_s belongs to v'_{KA}. Let's explore v'_{KA}."*
N_5	v_{KA}	v''_{KA}	v''_{KA} is subclass of v_{KA} and not superclass of v_s	*"You may find it useful to know that v''_{KA} belongs to v_{KA}. Let's explore v''_{KA}."*

Algorithm V describes our approach for generating exploration paths following Ausubels' subsumption theory for meaningful learning. The narrative types (N_1, N_2, … , N_5) used in this algorithm correspond to the narratives types described in Table 4 above.

The algorithm takes a data graph *DG*, the first entity v_s, the closest knowledge anchor v_{KA}, the length of the exploration path (*m*), and the edge label e = `rdfs:subClassOf` as an input, and generates an exploration path *P* of length *m*. The algorithm starts by initialising an empty exploration path *P* (line 1) used to store *m* transition narratives. If the first entity v_s is not the closest knowledge anchor v_{KA} (line 2), then the algorithm starts identifying the relationship type between v_s and v_{KA}. If v_s is subclass of v_{KA} (lines 3), then the following steps are conducted:

- The transition narrative $\langle v_s, script(N_1), v_{KA}\rangle$ from v_s to v_{KA} is inserted into *P* (line 4) where the function *script*(N_1) retrieves the *script output* for narrative type N_1 from Table 4. The length of exploration path *m* is decreased by one (line 5).
- The algorithm in (line 6) identifies the set of intermediate class entities (V'_{KA}) between v_s and v_{KA}, using the following SPARQL query:

```
SELECT distinct ?intermediate
  WHERE {
  vs rdfs:subClassOf ?intermediate.
  ?intermediate rdfs:subClassOf vKA .}
```

Algorithm V: Subsumption Using Closest KA_{DG}

Input: $DG=\langle V,E,T\rangle$, $v_s \in V$, $v_{KA} \in KA_{DG}$, m = *length of exploration path*, e = `rdfs:subClassOf`
Output: an exploration path *P* of length *m*.

1. $P := \{\}$; *//empty exploration path P*
2. **if** $v_s \neq v_{KA}$ **then** *//v_s is not v_{KA}*
3. **if** $\exists\langle v_s, e, v_{KA}\rangle$ **then** *//v_s is subclass of v_{KA}*
4. $P \leftarrow \langle v_s, script(N_1), v_{KA}\rangle$; *//insert narrative*
5. $m--$; *//reduce length of P by one*
6. V'_{KA} is all $v'_{KA} : \exists\langle v_s, e, v'_{KA}\rangle \wedge \exists\langle v'_{KA}, e, v_{KA}\rangle$;
7. $Q' := \{\}$;
8. $Q' \leftarrow sortDepth(V'_{KA})$;
9. **for** $(i := 1\ ;\ i \leq |Q'| \wedge\ m \geq 0\ ;\ i++)$
10. $P \leftarrow \langle v_{KA}, script(N_4), Q'[i]\rangle$; *//insert narrative*
11. $m--$; *//reduce length of P by one*
12. **end for;**
13. **else if** $\exists\langle v_{KA}, e, v_s\rangle$ **then**
14. $P \leftarrow \langle v_s, script(N_2), v_{KA}\rangle$;
15. $m--$; *//reduce length of P by one*
16. **else if** $\exists\langle v_s, e, v_i\rangle \wedge \exists\langle v_{KA}, e, v_i\rangle$ **then**
17. $P \leftarrow \langle v_s, script(N_3), v_{KA}\rangle$; *//insert narrative*
18. $m--$; *//reduce length of P by one*
19. **end if;**
20. **else if** $v_s = v_{KA}$ **then** *//v_s is v_{KA}*
21. V''_{KA} is all $v''_{KA} : \exists\langle v''_{KA}, e, v_{KA}\rangle \wedge v''_{KA} \notin Q'$;
22. $Q'' := \{\}$;
23. $Q'' \leftarrow sortDepthDensity(V''_{KA})$;
24. **for** $(j := 1\ ;\ j \leq |Q''| \wedge\ m \geq 0\ ;\ j++)$ **do**
25. $P \leftarrow \langle v_{KA}, script(N_5), Q''[j]\rangle$; *//insert narrative*
26. $m--$;
27. **end for;**
28. **end if;**

- A list Q' is created in (line 7), and the function *sortDepth*() sorts the class entities $v'_{KA} \in V'_{KA}$ based on their depths starting from the least depth class entity (i.e. direct subclass of v_{KA}) to highest depth in ascending order, and inserts the sorted class entities into list Q' (line 8). The function *sortDepth*() identifies the depth of a class entity v'_{KA} via the following SPARQL query:

```
SELECT (count(?intermediate)-1 as
?depth)
 WHERE {
   v'KA rdfs:subClassOf ?intermediate.
   ?intermediate rdfs:subClassOf r .}
```

where r is the root entity in the data graph.

- The algorithm in (lines 9 – 12) uses the closest v_{KA} to subsume the class entities in Q'. The transition narrativ $\langle v_{KA}, script(N_4), Q'[i] \rangle$ from v_{KA} to $Q'[i]$ (i.e. v'_{KA}) is inserted into P (line 10) where the function *script*(N_4) retrieves the *script output* for narrative type N_4 from Table 4. The length of the path m is decreased by one (line 11).

If v_s is superclass of v_{KA} (lines 13), then the transition narrative $\langle v_s, script(N_2), v_{KA} \rangle$ from v_s to v_{KA} is inserted into P (line 14) where the function *script*(N_2) retrieves the *script output* for narrative type N_2 from Table 4. The length m of the path P is decreased by one (line 15).

If v_s and v_{KA} are siblings (i.e. share same superclass) (line 16), then the transition narrative $\langle v_s, script(N_3), v_{KA} \rangle$ from v_s to v_{KA} is inserted into P (line 17) where the function *script*(N_3) retrieves the *script output* for narrative type N_3 from Table 4. The length m of the path P is decreased by one (line 18).

However, if the first entity v_s equals the closest knowledge anchor v_{KA} (line 20), then the following steps are conducted:

- Identify the set of subclass entities (V''_{KA}) of v_{KA} which do not belong to Q' (line 21). A list Q'' is created in (line 22), and the function $sortDepthDensity()$ sorts the class entities $v''_{KA} \in V''_{KA}$ based on two their depths and density, as the following steps:
  - Identify the depth of the class entities (similar to function *sortDepth*() described above).
  - Identify the density (using degree centrality) of the class entities based on number of subclasses.
  - Sort the class entity starting from the least depth (i.e. direct subclasses of v_{KA}) to highest depth in ascending order, and from highest density to least density (i.e. first sort using depth, if two or more entities are at the same depth, then sort these entities based on their density from highest to lowest). The sorted class entities are inserted into list Q'' (line 23).
- The algorithm in (lines 24–27) uses v_{KA} to subsume the class entities in Q''. The transition narrative $\langle v_{KA}, script(N_5), Q''[j] \rangle$ from v_{KA} to $Q''[j]$ (i.e. v''_{KA}) is inserted into P (line 25) where the function *script*(N_5) retrieves the *script output* for narrative type N_5 from Table 4. The length of exploration path m is decreased by one (line 26).

8. Evaluation of the Subsumption Algorithm

To evaluate the subsumption algorithm, we will conduct an experimental user study with users to examine the knowledge utility and users' exploration experience of the generated exploration paths. We will compare two conditions:

Experimental condition (*EC*): where users follow exploration paths generated using the subsumption algorithm;

Control condition (*CC*): where users carry out free exploration and they are free to select entities to visit.

Comparing both conditions a controlled task-driven experimental user study will be conducted with participants to examine the following hypotheses:

H1. Users who follow EC expand their domain knowledge.
H2. The expansion in the users' knowledge when following EC is higher than when following CC.
H3. The usability when EC is followed is higher than when CC is followed.

8.1. Experimental Condition Setting: Exploration Task Design

Designing exploration tasks for users is considered an important requirement for evaluating data exploration approaches [89]. A typical exploration task has to be generic (i.e. the scope of the task is broad and the user don't have specific information needs), realistic (i.e. real-life task that set in a familiar situation), discovery-oriented (i.e. users travel beyond what they know), open-ended (i.e. requires a significant amount of exploration, where open-endedness relates to uncertainty over the information available, or incomplete information on the nature of the search task), and set in an unfamiliar domain for the user [2,89,90]. In this work, we follow a two-step approach (similar to [89]) to design a data exploration task for the study participants. The approach involves: (i) Designing a task template that places the participant in a familiar situation which involves exploring multiple entities in an unfamiliar domain or topic (e.g. a researcher at a university that wants to write a research paper (familiar situation) about a new topic), and (ii) Identifying unfamiliar candidate

entities (e.g. find new research topic) in the domain that could be plugged into the task template.

Our aim was to design a generic task template that encourages layman users to seek knowledge in a domain unfamiliar to them. Therefore, we designed the task template in the context of a general knowledge quiz show where layman users need to acquire as much knowledge as they can. Inspired by the task templates in [89], the task template in Table was designed to suit the musical instrument domain.

Table 5. Task template used in the experimental user study

Task template
*"Imagine that you are a member of a team which will take part in a general knowledge quiz show. You have been asked to explore **two musical instruments** for **20** minutes in order to prepare a short presentation to describe to your team what you have learned about these **instruments**".*

The second step in designing data exploration task was to identify unfamiliar entities in the domain of the user study. For this we ran a questionnaire with users to identify the unfamiliar entities in the `String Instrument` and `Wind Instrument` class hierarchies in the MusicPinta data graph. These two class hierarchies have the richest class representation in terms of the number of classes and the hierarchy depth as discussed in Section 3.5.1, and have the highest number of knowledge anchors (9 anchors in the `String Instrument` class hierarchy and 10 anchors in the `Wind Instrument` class hierarchy – out of 24 anchors in MusicPinta data graph). We have extracted class entities at the bottom quartile of the two class hierarchies (note that the depth of the two class hierarchies is 7 – see Table 1, and entities of depth 6 or 7 are considered to be at the bottom quartile of the data graph). This is based on earlier Cognitive science studies acknowledging that layman users are not familiar with specific objects in a domain [91]. Overall 61 class entities from the `String Instrument` and `Wind Instrument` class hierarchies were used in the survey. The selected classes were randomised and distributed among twelve participants who are not experts in the musical instruments (the participants have limited knowledge about musical instruments and may have seen the instrument, and none of the participants had played any musical instruments).

The most unfamiliar instrument from each class hierarchy was `Biwa` (class hierarchy: `String Instrument`, origin: Japanese) and `Bansuri` (class hierarchy: `Wind Instrument`, origin: Indian).

8.2. Experimental Setup

Participants. 32 participants consisting of university students and professionals (24 students and 8 professionals[13]) were recruited on a voluntary basis (a compensation of £5 Amazon voucher was offered). Participants varied in age 18–45 (mean age is 30), and cultural background (1 Austrian, 9 British, 1 Chinese, 3 Greek, 1 Italian, 5 Jordanian, 1 Libyan, 2 Malaysian, 6 Nigerian, 1 Polish, 1 Romanian and 1 Saudi).

Method. Four online surveys[14] were run (which can be found here[15]). Every survey had 8 participants. Each participant was allocated one survey. Figure 8 shows the overall structure of the user study. Each participant explored both musical instruments (i.e. `Biwa`, `Bansuri`), where each of the instrument was allocated to an exploration strategy (*EC* or *CC*). The order of *EC* and *CC* was randomised to counter balance the impact on the results. Every participant session was *conducted separately* and observed by the author. All participants were asked to provide feedback before, during, and after the interaction with MusicPinta.

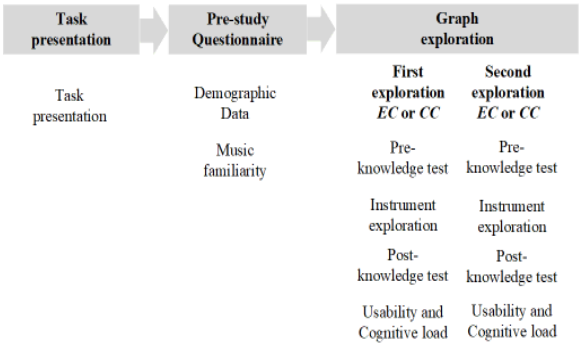

Fig. 8. Structure of user study to examine *EC* against *CC* in terms of knowledge utility, usability and cognitive load.

Task presentation [1 min] – utilise the task template (as described in Section 8.1) to present the data exploration task for the users at the beginning of their

[13] University lecturers and private Sector employees (Banking and Airlines).

[14] The study was conducted with Qualtrics (www.qualtrics.com).

[15] https://login.qualtrics.com/jfe/preview/SV_39nOZZWAmCk8Jp3?Q_CHL=preview

exploration session. We used the task template in Table 5 for Biwa and Bansuri.

Pre-study questionnaire [2 min] - collected information about the participants' profiles, and their familiarity with the music domain, focusing on the two musical instrument class hierarchies which would be explored – String Instrument and Wind Instrument. The participants' familiarity with the two class hierarchies varied from low to medium (63% and 78% of the participants had low familiarity with String Instrument and Wind Instrument, respectively).

Graph exploration [20 min] – each user explored the two strategies (*EC* and *CC*) where each strategy corresponds to one of the two unfamiliar instruments (Biwa or Bansuri). Figure 9 shows an example for generating the exploration path under *EC* for the musical instrument Biwa (the first entity of the exploration path) using the closest knowledge anchor Lute.

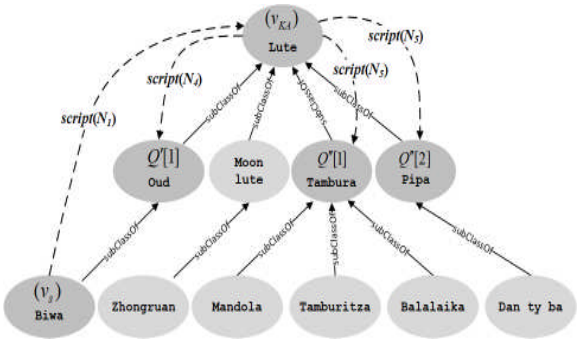

Fig. 9. Extract from the String Instrument class hierarchy in MusicPinta showing exploration path of Biwa. This path was followed in the experimental condition *EC*.

The first transition narrative in the exploration path is between the Biwa and the closest knowledge anchor Lute (using N_1 in Table 4). After that, the closest knowledge anchor Lute is used to subsume new class entities and generate transition narratives in the exploration path using the narrative types N_4 (subsume intermediate class entities between Lute and Biwa – line 10 in Algorithm V) and N_5 (subsume subclasses of Lute other than class entities that have been subsumed using N_4 – line 25 in Algorithm V).

The transition narratives that were followed in generating the exploration path for Biwa are listed in Table 6.

Table 6. Transition narratives used for generating the exploration path for Biwa

Transition Narratives for path of Biwa	Narrative Script
$\langle v_s, script(N_1), v_{KA} \rangle$	You may find it useful to know that '*Biwa*' belongs to a familiar and well-known instrument called '*Lute*'. Let's explore '*Lute*'.
$\langle v_{KA}, script(N_4), Q'[1] \rangle$	You may also find it useful to know that '*Oud*' belongs to '*Lute*' , and '*Biwa*' belongs to '*Oud*'. Let's explore '*Oud*'.
$\langle v_{KA}, script(N_5), Q''[1] \rangle$	You may also find it useful to know that '*Tambura*' belongs to '*Lute*'. Let's explore '*Tambura*'
$\langle v_{KA}, script(N_5), Q''[2] \rangle$	You may also find it useful to know that '*Pipa*' belongs to '*Lute*'. Let's explore '*Pipa*'

Table 7 shows examples of the entities that were freely visited in the control condition *CC* for Biwa.

Table 7. Examples of entities the participants have visited during their free exploration of Biwa

Example 1	Example 2	Example 3
Biwa	Biwa	Biwa
Bouzouki	String Instruments	Japanese Musical Instruments
Xalam	'Guitar'	Lute
Banjitar	Acoustic Guitar	Moon Lute
Plucked String instruments	Classical Guitar	Bouzouki

Figure 10 shows an example for generating the exploration path under *EC* for the instruments Bansuri (the first entity v_s in the exploration path) using the closest knowledge anchor Flute.

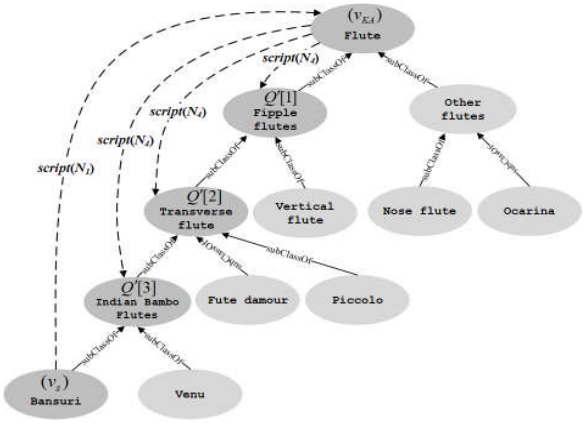

Fig. 10. Extract from the Wind Instrument class hierarchy in MusicPinta showing exploration path of Bansuri. This path was followed in experimental condition *EC*.

The exploration path for Bansuri was generated using the closest knowledge anchor Flute and nar-

rative types N_1, N_4 given in Table 4. The first transition narrative is between the first entity `Bansuri` and the closest knowledge anchor `Flute` (using N_1 in Table 4). After that, the closest knowledge anchor `Flute` is used to subsume new class entities and to generate transition narratives in the exploration path using narrative type N_4 (subsume intermediate class entities between `Flute` and `Bansuri` line 10 in Algorithm V). Transition narratives that were followed in generating the exploration path for `Bansuri` are listed in Table 8.

Table 8. Narrative scripts used for generating the experimental condition *EC* (i.e. exploration path) for instrument `Bansuri`

Transition Narratives for path of `Bansuri`	Narrative Script
$\langle v_s, script(N_1), v_{KA} \rangle$	*You may find it useful to know that 'Bansuri' belongs to a familiar and well-known instrument called 'Flute'. Let's explore 'Flute'.*
$\langle v_{KA}, script(N_4), Q'[1] \rangle$	*You may also find it useful to know that 'Fipple Flute' belongs to 'Flute' , and 'Bansuri' belongs to 'Fipple Flute'. Let's explore 'Fipple Flute'.*
$\langle v_{KA}, script(N_4), Q'[2] \rangle$	*You may also find it useful to know that 'Transverse Flute' belongs to 'Flute' , and 'Bansuri' belongs to 'Transverse Flute'. Let's explore 'Transverse Flute'.*
$\langle v_{KA}, script(N_4), Q'[3] \rangle$	*You may also find it useful to know that 'Indian Bamboo Flutes' belongs to 'Flute' , and 'Bansuri' belongs to 'Indian Bamboo Flutes'. Let's explore 'Indian Bamboo Flutes'.*

Table 9 shows examples of the entities that were freely visited in the control condition *CC* for `Bansuri`

Table 9. Examples of entities the participants have visited during their free exploration of `Bansuri`

Example 1	Example 2	Example 3
`Bansuri`	`Bansuri`	`Bansuri`
`Transverse Flute`	`Fipple Flute`	`Bamboo Musical Instruments`
`Saw Truck`	`Contrabass Recorder`	`Side-blown Flute`
`Fipple Flute`	`Recorder`	`Concert Flute`
`Flute D'amour`	`Great bass recorder`	`Fipple Flute`

Both, *EC* and the *CC* had the same length (*EC* had four transition narratives, and *CC* had four edges)[16]. We analysed *knowledge utility* and *user exploration experience* using usability aspects, associated with the user's exploration settings under the experimental condition (*EC*) and the control condition (*CC*), as the following:

Measuring knowledge utility. To compare the knowledge utility of an exploration path, we need a reliable approach for measuring knowledge. For this, we adopt the well-known taxonomy by Bloom used for assessing conceptual knowledge [92]. The taxonomy identifies a set of progressively complex learning objectives that can be used to design or assess learning experiences over information seeking and search tasks, and offers a means of assessing the depth of learning that occurs through search [93]. It suggests linking knowledge to six cognitive processes: remember, understand, apply, analyze, evaluate, and create. Among these, 'remember' and 'understand' are directly related to browsing and exploration activities. The remaining processes require deeper learning activities, which usually happen outside a tool, in our case data browser, and hence will not be considered. The 'remember' process is about retrieving relevant knowledge from the long-term memory, and includes recognition (locating the knowledge) and recall (retrieving it from the memory) [92]. The 'understand' process is about constructing meaning, from which the most relevant to a semantic browser is categorise (determining that an entity belongs to a particular category) and compare (detecting similarities) [92].

To measure the knowledge utility of a path, we use a schema activation technique used for assessing how users expand their knowledge after reading text [94]. To assess user's knowledge of a target domain concept, the user is asked to name concepts that belongs to and are similar to the concept. The schema activation test is conducted before an exploration and after an exploration, using three questions related to *the selected cognitive processes of* remember, categories, *and* compare:

- **Q1 [remember]** *What comes in your mind when you hear the word X?*;
- **Q2 [categorise]** *What musical instrument categories does X belong to?*;
- **Q3 [compare]** *What musical instruments are similar to X?*

The number of *accurate* concepts named (e.g. naming an entity with it's *exact name*, or with a *parent* or with a *member* of the entity) by user before and after exploration is counted, and the *difference indicates the knowledge utility of the exploration*. For example, if a user could name correctly *two* musical instruments similar to the musical instrument `Biwa` (Q3) before his/her exploration and then the user

[16] The length of four edges (5 entities) is based on Miller's Law [100], which indicates the number of objects that an average human can hold in working memory is 7 ± 2.

could name correctly *six* names of musical instruments similar to the instrument `Biwa` after his/her exploration, then the effect of the exploration on the cognitive process *compare* is indicated as 4 (i.e. as a result of the exploration the user learned 4 new similar musical instruments to the musical instrument `Biwa`). If a user named one instrument after his/her exploration (i.e. knowledge utility is -1), in such cases the knowledge utility equal to zero.

User's exploration experience of a path. After each exploration, the participants were asked to fill a questionnaire about their exploration experience and the cognitive load they have experienced (based on a modified version of the NASA-TLX questionnaire [95]). Furthermore, the participants were asked to think aloud and notes of all comments were kept.

8.3. Results

In the following, we present the results of evaluating the subsumption algorithm. We have analysed the *user knowledge expansion* and *user exploration experience* by usability aspects, associated with the user's exploration settings under the experimental condition (*EC*) and the control condition (*CC*).

8.3.1. Measuring Knowledge Utility

The user knowledge was measured before and after each exploration using the three questions of the schema activation test related to the entities `Biwa` and `Bansuri`. *Before exploration, none of the users were able to articulate any item linked to the two musical instruments (`Biwa` and `Bansuri`) using the three cognitive processes.* The knowledge utility for the three cognitive process before and after exploration of EC and CC is shown in Figure 11.

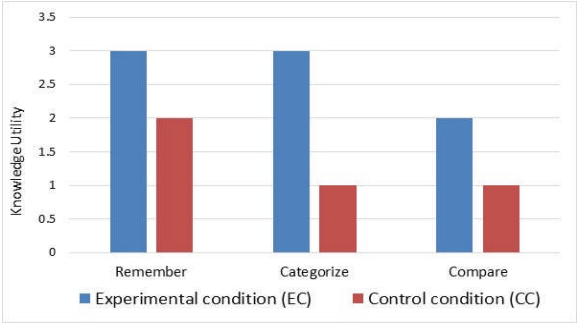

Fig. 11. Knowledge utility of the two strategies (*EC* and *CC*) of the user cognitive processes (*median* of the knowledge utility of exploration for all users).

The knowledge utility of the exploration under experimental condition (*EC*) in the three cognitive processes was higher than the effect of free exploration under control condition (*CC*); and this difference is significant (See Table 10). The results showed that all participants were able to *remember* and *categorise* entities with *EC* (only 5 participants couldn't *compare* new entities). Whereas not all participant could *remember*, *categorise* or *compare* new entities after they have finished their exploration with *CC* (there were 2 participants that could not *remember* or *categorise* new entities and 13 participants that could not compare between entities).

Table 10. Statistically significant differences of the values in Figure 11 (Mann-Whitney, 1-tail, Na=Nb=32)

Difference in Knowledge Utility between *P* and *F*	Cognitive Process	Z-value	p
EC > *CC*	Remember	3.6	P<0.01
	Categorise	5.1	P<0.0001
	Compare	2.7	P<0.01

Notably, for the cognitive process *categorise* the bigger effect on exploration of the subsumption exploration strategy over the free exploration strategy is highly significant (p<0.0001). The difference between median values for *categorise* cognitive process under *EC* and *CC* was higher than the *remember* and *compare* cognitive processes. Furthermore, we examined the knowledge utility for the three cognitive processes for each instrument in its corresponding class hierarchies, as shown in Figure 12.

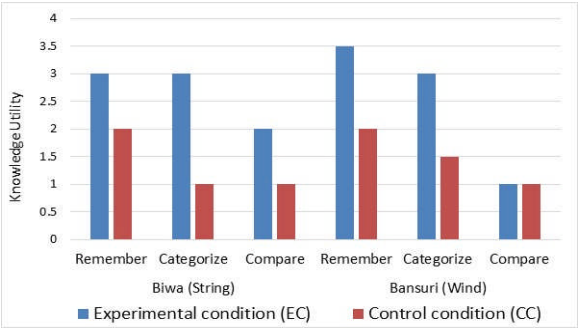

Fig. 12. Knowledge utility of the two strategies (*EC* and *CC*) of the user cognitive processes (*median* of the knowledge utility of exploration for all users).

The knowledge utility of the exploration under experimental condition (*EC*) in the three cognitive processes was higher than the effect of free exploration under control condition (*CC*) for `Biwa` and was higher in the cognitive processes *compare* and *categorise* for `Bansuri` (See Figure 12); and this difference in *EC* and *CC* is significant except for the cog-

nitive process *compare* for instrument `Bansuri` (See Table 11). By inspecting the participants' answers for the cognitive process *compare*, it was noticed that more than half of the participants (9 out of 16 participants for *EC*; 13 out of 16 participants for *CC*) had 0 or 1 values as their knowledge expansion in the cognitive process *compare* (0 value: participants didn't learn similar entities to `Bansuri`; 1 value: participants learned one entity similar to `Bansuri`). Since more than half of the results in the two lists (*EC* and *CC*) are 0 or 1, this decreases the difference between *EC* and *CC* (i.e. decrease the chance that a randomly selected value from *EC* will be higher than a randomly selected value from *CC*). Notably, for the cognitive process *categorise* the bigger effect on exploration of *EC* over *CC* is highly significant ($p<0.001$) for `Biwa` and `Bansuri`.

Table 11. Statistically significant differences of the values in Figure 12 (Mann-Whitney, 1-tail, Na=Nb=16)

Difference in Knowledge Utility between *EC* and *CC*	Instrument (*class Hierarchy*)	Cognitive Process	Z-value	p
EC > *CC*	`Biwa` (*String*)	Remember	1.658	P<0.05
		Categorise	3.373	P<0.001
		Compare	2.449	P<0.05
EC > *CC*	`Bansuri` (*Wind*)	Remember	3.467	P<0.001
		Categorise	3.900	P<0.001
		Compare	1.280	P<0.5

To further inspect what caused the low knowledge utility for the cognitive process *compare* for instrument `Bansuri`, we looked into the participants' familiarity with the `Wind Instrument` class hierarchy (the class hierarchy that `Bansuri` belongs to) and noticed that 78% of the participants had low familiarity (i.e. participants have limited knowledge and they may have seen some instruments) with `Wind Instrument`, whereas 65% of the participants had low familiarity with the `String Instrument` class hierarchy. One could argue that being more familiar with the `String Instrument` class hierarchy than the `Wind Instrument` class hierarchy, participants knew more entities to compare with. Furthermore, entities in the `String Instrument` class hierarchy are associated with more DBpedia categories compared to entities in the `Wind Instrument` class hierarchy (`String Instrument` has 255 and `Wind Instrument` has 161 DBpedia categories). Most of these categories are grouping musical instruments based on their cultural origin (e.g. `Chinese Musical Instruments`, `Japanese Musical instrument`, `Indian Musical Instruments`, `Greek Musical Instruments`), which helped the participants to associate entities from their cultures. This indicates important considerations of the data graph and the user familiarity for generating exploration paths for knowledge expansion.

8.3.2. User Exploration Experience

After each exploration strategy, the participants' feedback on the exploration experience was collected including exploration usability and exploration complexity, adapted from NASA-TLX [95] (Table 12).

Table 12. Questions to gather feedback on user exploration experience, adapted from NASA-TLX (mental demand, effort, performance).

Subjective process	Question text
Knowledge Expansion	*How much the exploration expanded your knowledge?*
Content Diversity	*How diverse was the content you have explored?*
Mental Demand	*How mentally demanding was this exploration?*
Effort	*How hard did you have to work in this exploration?*
Performance	*How successful do you think you were in this exploration?*

Figures 13 and 14 give a summary of the users' feedback.

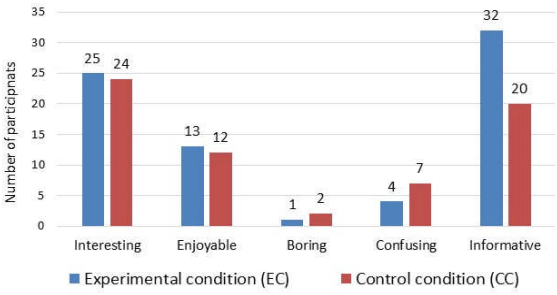

Fig. 13. Users' exploration experience of the two exploration strategies (*EC* and *CC*). Values show number of user paths rated with the corresponding characteristics.

In Figure 13, the exploration experience with *EC* was the most informative (all 32 participants identified their exploration experience with the exploration paths under *EC* as informative, whereas 20 participants indicated their exploration with *CC* as informative). The participants also found their exploration under *EC* to be slightly more interesting and enjoyable than *CC*. Furthermore, the participants found the

exploration paths under *EC* to be the least boring and least confusing – only 3% (one participant) and 16% (four participants) of the participants founded their exploration with *EC* to be boring or confusing, respectively. For instance, on participant indicated his exploration experience with *EC* as boring since he was not able to freely explore through entities in the graph (his feedback was "*Narratives in paths allows me to explore entities in a hierarchical fashion, and I would like to freely explore other types of relationships*"). Another participant indicated his exploration experience with *EC* as confusing (his feedback was "*I saw the same instruments several times during my exploration*".

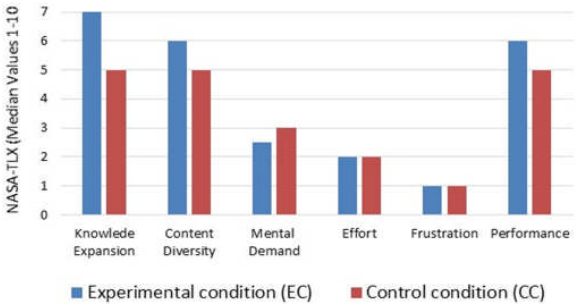

Fig. 14. Users' subjective perception of the two exploration strategies (*EC* and *CC*), based on an adapted NASA-TLX questionnaire [96] (median values for all users in the range 1-10).

The effect of the exploration path *under EC* on the subjective processes *knowledge expansion* and *performance* was higher than the effect of the free exploration strategy; and this difference was significant (See Table 13 and Fig 14).

Table 13. Statistically significant differences of users' subjective perception of cognitive process (Mann-Whitney, 1-tail, Na=Nb=32)

Difference in the users' experience	Subjective Process	Z value	p
EC > *CC*	Knowledge Expansion	3.98	P<0.0001
	Content Diversity	0.32	P<0.5
	Mental Demand	0.14	P<0.5
	Effort	0.14	P<0.5
	Performance	2.15	P<0.05

Notably, for the subjective process *knowledge expansion* the bigger effect on exploration of *EC* over *CC* is highly significant (p<0.0001).

9. Discussion

In this section, we revisit our research questions related to finding computational ways to: (i) identify knowledge anchors KA_{DG} and (ii) generate exploration paths for knowledge expansion using KA_{DG}. We reflect on how and to what extent we have answered them, discuss the major contributions of our work, and point at limitations.

9.1. RQ1: Identification of KA_{DG}

To address the first research question, we utilised Rosch's definitions of basic level objects and cue validity to develop two groups of metrics and the corresponding algorithms for identifying knowledge anchors in a data graph. These include: *distinctiveness* metrics, which identify the most differentiated categories whose attributes are shared amongst the category members but are not associated to members of other categories, and *homogeneity* metrics, which identify categories whose members share many attributes together. The formal framework that maps Rosch's definitions of BLO and cue validity to data graphs is a major contribution of our work.

9.1.1. KA_{DG} Algorithms

The KA_{DG} algorithms have been formally defined; they are generic and can be applied over different application domains represented as data graphs. The algorithms have been applied over two data graphs – in music (presented here) and in careers (presented in [97]). Although this paper focuses only on the music domain (so that we can show a holistic approach illustrated with a concrete application), in the discussion below we identify key features of the algorithms which have been confirmed in broader evaluation (see [97] for further detail).

The implementation showed that the output of the KA_{DG} algorithms is sensitive to the quality of the data graph in terms of the richness of the class entities and the hierarchy depth. Specifically, the algorithms tend to pick more anchoring entities when a data graph has many classes and high depth. For instance, the algorithms identified 9 anchors in the `String Instrument` class hierarchy and 10 anchors in the `Wind Instrument` class hierarchy – out of 24 anchors (`String Instrument` and `Wind Instrument` are the richest class hierarchies in MusicPinta – See table 1). Whereas the algorithms did not produce any anchor in the `Electronic Instrument` class hierarchy (only 15 classes with a depth of one). The same observation was confirmed in the career domain [97].

9.1.2. Identifying Human BLO_{DG}

To enable objective comparison of the outputs by the KA_{DG} algorithms and the cognitive structures of humans, we presented an algorithm that adapts earlier Cognitive Science experimental approaches of free-naming tasks to capture human basic level objects in a data graph (BLO_{DG}) that correspond to familiar concepts in human cognitive structures over a data graph. A user study in the music domain was conducted to identify benchmarking sets of human BLO_{DG} used to examine the performance of the KA_{DG} metrics. Based on quantitative and qualitative analysis, the strengths and limitations of each metric were assessed, and a hybridisation to enhance the performance of the metrics approach was proposed.

An important component for applying the human BLO_{DG} algorithm is to identify appropriate stimuli to be used for representing graph entities and showing them to humans in a free-naming tasks. One of the main factors that affects choosing appropriate stimuli is how well the stimuli cover the entities in the data graph. In other words, the chosen stimuli should have representations for all entities in the graph ontology. For instance, the stimuli for MusicPinta were images – taken from an established source (MIMO[7]). The chosen stimuli have to be close enough to users' cognitive structures, so the users can understand the representation of entities.

The formal description of the human BLO_{DG} algorithm makes it applicable over a broad range of domains, by selecting appropriate stimuli. In this paper we presented an application in a domain with concrete objects - musical instruments - that can have digital representations (e.g. image, audio, video). In [97] we have shown how the algorithm can be applied also to a data graph in an abstract domain (careers) where domain objects have text representations (i.e. labels of entities) but no clearly distinguishable digital representations.

The derived human BLO_{DG} set is depends on what categories are represented in the data graph. If key domain concepts, which are well-familiar to people, are missing in the data graph, they cannot be shown to the users, and hence will be missing from the derived human BLO_{DG} set. This is important for navigation through the data graph, as the evaluation will consider only entities that are in the data graph (and hence can be used in navigation paths). Furthermore, the derived human BLO_{DG} can point at deficiencies of the data graph. For instance, if a well-known domain category is not present in the derived BLO_{DG} this can indicate that the category is unrepresented in the data graph. Such findings can be useful for ontology evaluation and re-engineering.

9.1.3. Evaluating KA_{DG} Against Human BLO_{DG}

An important step in the evaluation is to identify a suitable cut-off point for the KA_{DG} metrics. In the current implementation, this was done through experimentation – the best performance was by using the 60^{th} percentile. The same percentile was identified as best for the career domain (see [87]). We expect that when applied to a range of data graphs, the 60^{th} percentile will give reasonable performance (the best cut-off point for a specific data graph would require experimentation comparing different percentiles).

The analysis indicated that hybridisation of the metrics notably improved performance. The same was observed in the careers domain ([97]). Appropriate hybridisation heuristics for the upper level of the data graph is to combine the KA_{DB} metrics using majority voting. The hybridisation heuristics for the bottom level of the hierarchy are dependent on the domain-specific relationships in the data graph. Hence, to derive appropriate hybridisation heuristics that give good performance for categories at the bottom level, further experimentation will be required. This will include comparing the KA_{DB} derived using the various domain-specific relationships against human BLO_{DB}.

9.2. RQ2: Generation of Exploration Paths in Data Graphs

9.2.1. Subsumption Algorithms for Generating Paths

We adopted the subsumption theory for meaningful learning [21] to generate exploration paths for knowledge expansion using KA_{DG}. For this, we formally described two algorithms.

Algorithm for identifying the closest knowledge anchor to the first entity of an exploration path. It applies a semantic similarity metric based on class hierarchy depth to identify the closest knowledge anchor to the first entity of an exploration path. This similarity algorithm is not applicable in data graphs with shallow class hierarchies (e.g. class hierarchy depth = 1 or 2). In such cases, there will be no common ancestors for the *first entity* and the *knowledge anchors*, and hence the semantic similarity value will be zero. Furthermore, two (or more) knowledge anchors in a data graph can have the same semantic similarity value with the first entity. For example, there were two knowledge anchors (`Flute` and `Reeds`) with the same semantic similarity value with

the first entity `Bansuri`. One possible way to address this is to filter the knowledge anchors based on their density and give preference to the densest anchor as it will include many subclass members to subsume while generating the exploration path.

In some cases, the semantic similarity metric can identify closest knowledge anchors which are not superclass (i.e. N_1 in Table 4), subclass (i.e. N_2 in Table 4) nor sibling (i.e. N_3 in Table 4) to the first entity, which means that the closest knowledge anchor can not be reached directly from the first entity using a narrative transition. For instance, although the anchors `Flute` and `Reeds` have the same semantic similarity value with the first entity `Bansuri`, `Flute` is a superclass of `Bansuri` that can be reached directly using the narrative type (N_1 in Table 4), whereas `Reeds` can't be reached directly from `Bansuri`. Our solution to address this issue was to apply semantic similarity to knowledge anchors which could be reached directly from the first entity. Another solution can be to add a narrative type in Table 4 that indicates that the first entity and the closest knowledge anchor simply belong to the same domain but there is no direct trajectory between them.

Algorithm for generating exploration path as a set of transition narratives using the closest knowledge anchor. The algorithm is underpinned by the subsumption theory for meaningful learning to generate exploration paths for knowledge expansion. The algorithm uses a knowledge anchor to subsume subordinate entities (i.e. subclasses of the closest knowledge anchor) while generating transition narratives of an exploration path. The algorithm is dependent on the sub-classes of the selected knowledge anchor – it may not be possible to generate *m* transition narratives in the exploration path (where *m* is the required length of exploration path identified as an input of the algorithm). Our implementation uses only the knowledge anchor, and can result in paths whose length of less than *m*. Another way to address this would be to continue the path, using another knowledge anchor. It is also possible to use superordinate categories to the knowledge anchor to extend an exploration path. This will be suitable for cases when the user is gained knowledge at the subordinate level and is ready to generalise to a more abstract level. In such cases, a user model will be required.

9.2.2. Evaluation of Exploration Paths

Overall, the evaluation results have supported our hypothesis *H1-H3 as set out in Section 8.*

H1: When users followed the experimental condition, i.e. the paths generated by the subsumption algorithm, they expanded their domain knowledge. All participants in the study have indicated that their exploration in the experimental condition was informative (in other words, they felt that while following the path they were able to find useful information); whereas 62% of the participants founded their free exploration trajectories to be informative.

H2: The expansion of the users' knowledge when following the generated exploration paths was higher than when following free exploration - the participants were able to *remember*, *categorize* and *compare* significantly more entities. The results also showed that the cognitive process *categorise* had the bigger effect on expanding the participants' knowledge. This was caused because: (i) the subsumption hierarchical relationship (`rdfs:subClassOf`) was used to create the narrative scripts between entities of the generated exploration paths, which helped the users to categorise new entities at different levels of abstraction at their cognitive structures; and (ii) the subsumption process uses knowledge anchors to subsume and learn new subcategories similar to the way a human mind works while learning a new concept.

H3: The results showed that participants found the generated exploration paths to be more enjoyable and less confusing than free exploration paths, and their assessment of performance was higher. One participant founded his experience with hierarchical narrative scrips to be boring; and suggested that the system should diversify the types of narratives used between entities in the exploration path. For example, to use other relationships, other than the subsumption relationship, for generating transition narratives.

9.2.3. Applicability of Evaluation Approach

Instruments used in the evaluation of the exploration paths can be applied in other exploratory search evaluation studies.

To measure knowledge utility of an exploration path, the user knowledge is assessed before and after exploration using Bloom's cognitive processes of *remember*, *categorise* and *compare*. These were extracted from the first two cognitive categories in Bloom's taxonomy (namely *remember* and *understand*) which are directly related to exploration activities. In other evaluation contexts, more complex cognitive categories can be applied such as the cognitive category *analyse*. This category includes several cognitive processes, such as *differentiate* (e.g. differ-

entiate between two entities in the data graph) and *arrange* (e.g. arrange entities in the data graph from abstract to specific), which can be related to exploratory search tasks.

We evaluated the subsumption algorithm for generating exploration paths against free exploration by adopting a task-based approach. It involved two steps: (i) designing a task template and (ii) identifying unfamiliar entities in the domain to be plugged into the task template. The task template presented in Section 8 is in the context of a general knowledge quiz that encourages users to seek knowledge in a given domain represented as a data graph. The template can easily be adapted for a range of data graph exploration tasks. This will require identifying unfamiliar entities to include in the template. We did this based on a small survey with participants to identify domain entities (at the bottom quartile of the data graph) which are likely to be unfamiliar to layman user. At a larger scale, crowdsourcing can be used to identify unfamiliar entities in the data graph.

10. Conclusion and Future Work

Exploration of data to carry out an open-ended task is becoming a key daily life activity. It usually involves a journey through large datasets or search systems that starts with an entry point, often an initial query, and then exploring a large amount of data while constantly making decisions about which data to explore next. Users who are unfamiliar with the domain they are exploring can face high cognitive load and usability challenges when exploring such large amount of data. Our work investigates how to support such users' exploration through a data graph in a way that leads to expansion of the user's domain knowledge. We introduced a novel exploration support mechanism underpinned by the subsumption theory of meaningful learning, which postulates that new knowledge is grasped by starting from familiar concepts in the graph which serve as knowledge anchors from where links to new knowledge are made.

The KA_{DG} algorithms can have several applications. In *data graph exploration*, KA_{DG} enables operationalising the subsumption theory for meaningful learning [21] to generate exploration paths for knowledge expansion. Furthermore, our approach for identifying KA_{DG} can be applied to *ontology summarisation* [53] where KA_{DG} allow capturing a layman view of the domain. KA_{DG} approach can also be applied to solve the 'cold start' problem in *personalization and adaptation* [98]. One of the popular choices for addressing the cold start problem is a dialogue system with the user. The data graphs can provide a large knowledge pool to implement such probing dialogues, however, one needs to select entities from the vast amount of possibilities for probing to avoid too long interactions with the user. Knowledge anchors can be used as the starting entities for the probing dialogues [99].

The subsumption algorithm is generic and can be applied to generate transition narratives for exploratory search in other domains. Further extension would be to maintain a user model and personalise the path to the individual's domain knowledge. Another future direction would be to add a path diversification strategy to suggest interesting concepts that are more likely to attract people to engage in the exploration, and hence learn more about the domain.

Acknowledgements. This research was supported by a Doctoral Grant from the University of Jordan. The MusicPinta semantic data browser was developed by the EU/FP7 project Dicode. We are grateful to all participants in the experimental studies.

References

[1] E. Palagi, F. Gandon, A. Giboin, R. Troncy, I.S. Antipolis, F. Gandon, I.S. Antipolis, A. Giboin, and I.S. Antipolis, A survey of definitions and models of exploratory search, in: ACM Work. Explor. Search Interact. Data Anal. - ESIDA '17, 2017: pp. 3–8. doi:10.1145/3038462.3038465.

[2] R.W.W. and R.A. Roth, Exploratory Search Beyond the Query–Response Paradigm, Morgan & Claypool, 2009. doi:10.1287/orsc.1110.0676.

[3] N. Belkin, Anomalous States of Knowledge as the Basis of Information Retrieval., *Can. J. Inf. Sci.* **5** (1980) 133–143.

[4] G. Marchionini, Exploratory search: from finding to understanding, in: Commun. ACM, 2006: p. 41. doi:10.1145/1121949.1121979.

[5] T. Berners-lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, Tabulator : Exploring and Analyzing linked data on the Semantic Web, in: L. Rutledge, M C Schraefel, A. Bernstein, and D. Degler (Eds.), 3rd Int. Semant. Web User Interact. Work., Citeseer, 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.950&rep=rep1&type=pdf.

[6] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellamnn, DBpedia - A cystallization point for the Web of Data, *Web Semant. Sci. Serv. Agents World Wide Web*. **7** (2009) 154–165.

[7] G. Cheng, Y. Zhang, and Y. Qu, Explass: Exploring Associations between Entities via Top-K Ontological Patterns and Facets, in: ISWC '13, 2014: pp. 422–437. doi:10.1007/978-3-319-11915-1_27.

[8] G.C. and Y. Qu, Searching linked objects with falcons: Approach, implementation and evaluation, *Int. J. Semant.*

Web Inf. Syst. **5** (2009) 49–70.
[9] L. Zheng, Y. Qu, J. Jiang, and G. Cheng, Facilitating entity navigation through top-K link patterns, in: Proc. Int. Semant. Web Conf., 2015: pp. 163–179. doi:10.1007/978-3-319-25007-6_10.
[10] M. Sah, and V. Wade, Personalized concept-based search on the Linked Open Data, *Web Semant. Sci. Serv. Agents World Wide Web*. **36** (2016) 32–57. doi:10.1016/j.websem.2015.11.004.
[11] L. Zheng, Y. Qu, and G. Cheng, Leveraging link pattern for entity-centric exploration over Linked Data, in: Proc. WWW, World Wide Web, 2017. doi:10.1007/s11280-017-0464-y.
[12] R. Pienta, G. Tech, J. Vreeken, G. Tech, and J. Abello, FACETS : Adaptive Local Exploration of Large Graphs, in: IEEE SDM'17, 2017.
[13] S. Dietze, and E. Kaldoudi, Socio-semantic Integration of Educational Resources - the Case of the mEducator Project, *Univers. Comput. Sci.* **19** (2013) 1543–1569.
[14] P. Vakkari, Searching as learning: A systematization based on literature, *J. Inf. Sci.* **42** (2016) 7–18. doi:10.1177/0165551515615833.
[15] J. Gwizdka, P. Hansen, C. Hauff, J. He, and N. Kando, Search As Learning (SAL) Workshop 2016, in: Proc. 39th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr., ACM, New York, NY, USA, 2016: pp. 1249–1250. doi:10.1145/2911451.2917766.
[16] L. Koesten, E. Kacprzak, and J. Tennison, Learning When Searching for Web Data., in: Sal@SigirSearch as Learn., 2016: pp. 2–3. http://dblp.uni-trier.de/db/conf/sigir/sal2016.html#KoestenKT16.
[17] V. Dimitrova, L. Lau, D. Thakker, F. Yang-Turner, and D. Despotakis, Exploring exploratory search: a user study with linked semantic data, *Proc. 2nd Int. Work. Intell. Explor. Semant. Data - IESD '13*. (2013) 1–8. doi:978-1-4503-2006-1.
[18] D. Thakker, V. Dimitrova, L. Lau, F. Yang-Turner, and D. Despotakis, Assisting user browsing over linked data: Requirements elicitation with a user study, in: ICWE'13 Int. Conf. Web Eng., 2013: pp. 376–383. doi:10.1007/978-3-642-39200-9_31.
[19] V. Maccatrozzo, Burst the filter bubble: using semantic web to enable serendipity, in: ISWC '11, 2012. doi:10.1007/978-3-642-35173-0.
[20] M. Al-Tawil, D. Thakker, and V. Dimitrova, Nudging to Expand User ' s Domain Knowledge while Exploring Linked Data, in: IESD@ISWC, 2014.
[21] D.P. Ausubel, A subsumption theory of meaningful verbal learning and retention, *J. Gen. Psychol.* **66** (1962) 213–224. doi:10.1080/00221309.1962.9711837.
[22] E. Rosch, C.B. Mervis, W.D. Gray, D.M. Johnson, and P.Boyes-Braem, Basic Objects in Neutral categories, *Cogn. Psychol.* **8** (1976) 382–439.
[23] S. Mazumdar, D. Petrelli, K. Elbedweihy, V. Lanfranchi, and F. Ciravegna, Affective graphs: The visual appeal of Linked Data, *Semant. Web.* **6** (2015) 277–312. doi:10.3233/SW-140162.
[24] B. Fu, N.F. Noy, and M. Storey, Eye Tracking the User Experience - An Evaluation of Ontology Visualization Techniques, *Semant. Web J.* **8** (2017) 23–41. http://www.semantic-web-journal.net/system/files/swj770.pdf.
[25] A.S. Dadzie, and E. Pietriga, Visualisation of Linked Data - Reprise, *Semant. Web*. **8** (2017) 1–21. doi:10.3233/SW-160249.
[26] M. Javed, S. Payette, J. Blake, and T. Worrall, VIZ – VIVO : Towards Visualizations-driven Linked Data Navigation, in: VOILA@ISWC, 2016.
[27] I.O. Popov, M.C. Schraefel, W. Hall, and N. Shadbolt, Connecting the Dots: A Multi-pivot Approach to Data Exploration, in: ISWC'10, 2011: pp. 553–568. doi:http://dx.doi.org/10.1007/978-3-642-25073-6_35.
[28] S. Araújo, D. Schwabe, and S.D.J. Barbosa, Experimenting with Explorator: A direct manipulation generic RDF browser and querying tool, in: VISSW, 2009.
[29] D. Huynh, and D. Karger, Parallax and companion: Set-based browsing for the data web, *WWW Conf.* (2009) 2005–2008. http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf.
[30] G. Vega-Gorgojo, M. Giese, S. Heggestøyl, A. Soylu, and A. Waaler, PepeSearch: semantic data for the masses, *PLoS One*. **11** (2016).
[31] B. Shneiderman, The eyes have it: a task by data type taxonomy for information visualizations, in: Proc. 1996 IEEE Symp. Vis. Lang., IEEE, 1996: pp. 336--343. http://reference.kfupm.edu.sa/content/e/y/the_eyes_have_it__a_task_by_data_type_ta_43453.pdf.
[32] A. Valsecchi, M. Abrate, C. Bacciu, M. Tesconi, and A. Marchetti, Linked Data Maps: Providing a Visual Entry Point for the Exploration of Datasets, *IESD@ISWC*. **1472** (2015).
[33] P.R. Smart, A. Russell, D. Braines, Y. Kalfoglou, J. Bao, and N. Shadbolt, A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer, in: Proc. 16th Int. Conf. Knowl. Eng. Knowl. Manag., 2008: pp. 275–291. doi:10.1007/978-3-540-87696-0_25.
[34] M. Zviedris, and G. Barzdins, Viziquer: A tool to explore and query SPARQL endpoints, in: Proc. 8th ESWC, 2011: pp. 441–445. doi:10.1007/978-3-642-21064-8.
[35] W. Javed, S. Ghani, and N. Elmqvist, PolyZoom : Multiscale and Multifocus Exploration in 2D Visual Spaces, in: CHI'12, ACM, 2012. http://dl.acm.org/citation.cfm?id=2207716.
[36] S. Scheider, A. Degbelo, R. Lemmens, C. Van Elzakker, P. Zimmerhof, N. Kostic, J. Jones, and G. Banhatti, Exploratory querying of SPARQL endpoints in space and time, *Semant. Web*. **8** (2017) 65–86. doi:10.3233/SW-150211.
[37] A.G. Nuzzolese, V. Presutti, A. Gangemi, S. Peroni, and P. Ciancarini, Aemoo: Linked Data exploration based on Knowledge Patterns, *Semant. Web*. **8** (2017) 87–112. doi:10.3233/SW-160222.
[38] A. Harth, Visinav: Visual web data search and navigation, in: DEXA, 2009: pp. 214–228.
[39] P. Heim, T. Ertl, and J. Ziegler, Facet graphs: complex semantic querying made easy, in: L. Aroyo, G. Antoniou, E. Hyvönen, A. Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache (Eds.), ESWC'7th, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 288–302. http://dl.acm.org/citation.cfm?id=2176871.2176894 (accessed February 26, 2013).
[40] P. Heim, J. Ziegler, and S. Lohmann, GFacet: A browser for the web of data, in: IMC-SSW'08, 2008: pp. 49–58. doi:10.1.1.143.1026.
[41] S. Brunk, and P. Heim, Tfacet: Hierarchical faceted exploration of semantic data using well-known interaction concepts, in: DCI@ INTERACT, 2011: pp. 31–36.
[42] J.M. Brunetti, R. García, and S. Auer, From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data, *Int. J. Semant. Web Inf. Syst.* **9** (2013) 1–20. http://www.igi-global.com/article/overview-facets-pivoting-interactive-exploration/77822 (accessed January 13, 2014).

[43] C. Stadler, M. Martin, and S. Auer, Exploring the web of spatial data with facete, *Proc. 23rd Int. Conf. World Wide Web - WWW '14 Companion*. (2014) 175–178. doi:10.1145/2567948.2577022.

[44] P. Papadakos, and Y. Tzitzikas, Hippalus: Preference-enriched faceted exploration, in: EDBT@ICDT, 2014.

[45] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations, *IEEE Trans. Vis. Comput. Graph.* **22** (2016) 649–658. doi:10.1109/TVCG.2015.2467191.

[46] N. Bikakis, G. Papastefanatos, M. Skourla, and T. Sellis, A Hierarchical Framework for Efficient Multilevel Visual Exploration and Analysis, *SWJ*. **8** (2017) 139–179. http://arxiv.org/abs/1511.04750.

[47] M. Sah, and V. Wade, Personalized Concept-Based Search and Exploration on the Web of Data Using Results Categorization, in: Proc. Ext. Semant. Web Conf., 2013: pp. 532–547.

[48] O. Rossel, Implemention of a " search and browse " scenario for the LinkedData, in: IESD@ISWC, 2014.

[49] L. De Vocht, A. Dimou, J. Breuer, M. Van Compernolle, R. Verborgh, E. Mannens, P. Mechant, and R. Van De Walle, A visual exploration workflow as enabler for the exploitation of linked open data, in: IESD@ISWC, 2014.

[50] M. Dojchinovski, and T. Vitvar, Personalised Access to Linked Data, in: Knowl. Eng. Knowl. Manag., 2014: pp. 121–136. doi:10.1007/978-3-319-13704-9_10.

[51] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro, Semantics-aware Graph-based Recommender Systems Exploiting Linked Open Data, in: UMAP '16, 2016: pp. 229–237. doi:10.1145/2930238.2930249.

[52] F. Bianchi, M. Palmonari, M. Cremaschi, and E. Fersini, Actively Learning to Rank Semantic Associations for Personalized Contextual Exploration of Knowledge Graphs, in: Proc. 14th Int. Conf. ESWC 2017, 2017: pp. 120–135.

[53] X. Zhang, G. Cheng, and Y. Qu, Ontology summarization based on rdf sentence graph, in: Proc. 16th Int. Conf. World Wide Web WWW 07, ACM Press, 2007. http://portal.acm.org/citation.cfm?doid=1242572.1242668.

[54] R. Wille, Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies, *Form. Concept Anal.* (2005) 1–33. doi:10.1007/11528784_1.

[55] N. Li, and E. Motta, Evaluations of user-driven ontology summarization, *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*. **6317** (2010) 544–553. doi:10.1007/978-3-642-16438-5.

[56] N. Li, and E. Motta, Evaluations of user-driven ontology summarization, *EKAW 17th Int. Conf. Knowl. Eng. Manag. by Masses*. **6317** (2011) 544–553. doi:10.1007/978-3-642-16438-5.

[57] K. Wang, Z. Wang, R. Topor, J.Z. Pan, and G. Antoniou, Eliminating concepts and roles from ontologies in expressive descriptive logics, *Comput. Intell.* **30** (2014) 205–232. doi:10.1111/j.1467-8640.2012.00442.x.

[58] G. Troullinou, H. Kondylakis, E. Daskalaki, D. Plexousakis, F. Gandon, M. Sabou, and H. Sack, Ontology understanding without tears: The-summarization approach, *Semant. Web*. **8** (2017) 797–815. doi:10.3233/SW-170264.

[59] G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis, RDF Digest : Efficient Summarization of RDF / S KBs, in: Gandon F., Sabou M., Sack H., d'Amato C., Cudré-Mauroux P., Zimmermann A. Semant. Web. Latest Adv. New Domains. ESWC 2015. Lect. Notes Comput. Sci. Vol 9088. Springer, Cham, 2015. doi:10.1007/978-3-319-18818-8.

[60] S. Peroni, E. Motta, and M. Aquin, Identifying key concepts in an ontology , through the integration of cognitive principles with statistical and topological measures, in: ASWC '08, 2008.

[61] Y. Cai, W.H. Chen, H.F. Leung, Q. Li, H. Xie, R.Y.K. Lau, H. Min, and F.L. Wang, Context-aware ontologies generation with basic level concepts from collaborative tags, *Neurocomputing*. **208** (2016) 25–38. doi:10.1016/j.neucom.2016.02.070.

[62] E. Rosch, and B.B. Lloyd, Cognition and Categorization, *Lloydia Cincinnati*. **pp** (1978) 27–48. http://www.worldcat.org/title/cognition-and-categorization/oclc/3844382&referer=brief_results.

[63] J. Poelmans, D.I. Ignatov, S.O. Kuznetsov, and G. Dedene, Formal concept analysis in knowledge processing: A survey on applications, *Expert Syst. Appl.* **40** (2013) 6538–6560. doi:10.1016/j.eswa.2013.05.009.

[64] P. Cimiano, A. Hotho, and S. Staab, Learning Concept Hierarchies from Text Corpora Using Formal Concept Analysis, *J. Artif. Int. Res.* **24** (2005) 305–339. http://dl.acm.org/citation.cfm?id=1622519.1622528.

[65] W.C. Cho, and D. Richards, Improvement of precision and recall for information retrieval in a narrow domain: Reuse of concepts by formal concept analysis, in: Proc. - IEEE/WIC/ACM Int. Conf. Web Intell. WI 2004, 2004: pp. 370–376. doi:10.1109/WI.2004.10082.

[66] D. Richards, Ad-Hoc and Personal Ontologies: A Prototyping Approach to Ontology Engineering, in: PKAW, 2006: pp. 13–24.

[67] B. Sertkaya, OntoComP: A Protégé Plugin for Completing OWL Ontologies, in: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl (Eds.), Semant. Web Res. Appl. 6th Eur. Semant. Web Conf. ESWC 2009 Heraklion, Crete, Greece, May 31--June 4, 2009 Proc., Springer Berlin Heidelberg, Berlin, Heidelberg, 2009: pp. 898–902. doi:10.1007/978-3-642-02121-3_78.

[68] R. Belohlavek, and M. Trnecka, Basic level of concepts in formal concept analysis, *ICFCA'10*. **7278 LNAI** (2012) 28–44. doi:10.1007/978-3-642-29892-9_9.

[69] R. Belohlavek, and M. Trnecka, Basic level in formal concept analysis: Interesting concepts and psychological ramifications, *IJCAI Int. Jt. Conf. Artif. Intell.* (2013) 1233–1239.

[70] A. Bonifati, R. Ciucanu, and A. Lemay, Learning Path Queries on Graph Databases, in: Proc. 18th Int. Conf. Extending Database Technol., 2015. doi:10.5441/002/edbt.2015.11.

[71] K. Anyanwu, and A. Sheth, ρ-Queries Enabling Querying for Semantic Associationson the Semantic Web.pdf, in: Proc. 12th Int. Conf. World Wide Web, 2003: pp. 690–699. doi:10.1145/775152.775249.

[72] P. Heim, S. Hellmann, J. Lehmann, and S. Lohmann, RelFinder : Revealing Relationships in RDF Knowledge Bases, (n.d.).

[73] R. García, R. Gil, J.M. Gimeno, E. Bakke, and D.R. Karger, BESDUI: A benchmark for end-user structured data user interfaces, in: ISWC '16th, 2016: pp. 65–79. doi:10.1007/978-3-319-46547-0_8.

[74] D. Lamprecht, M. Strohmaier, D. Helic, C. Nyulas, T. Tudorache, N.F. Noy, and M.A. Musen, Using ontologies to model human navigation behavior in information networks: A study based on Wikipedia, *Semant. Web*. **6** (2015) 403–422. doi:10.3233/SW-140143.

[75] D.P. Ausubel, and D. Fitzgerald, ANTECEDENT LEARNING VARIABLES IN SEQUENTIAL VERBAL

LEARNING, (1962).
[76] D.P. Ausubel, The use of advance organizers in the learning and retention of meaningful verbal material, *J. Educ. Psychol.* **51** (1960) 267–272. doi:10.1037/h0046669.
[77] D.P. Ausubel, Cognitive structure and the facilitation of meaningful verbal learning, *J. Teach. Educ.* **14** (1963) 217–222. doi:10.1177/002248716301400220.
[78] D.P. Ausubel, J.D. Novak, and Helen Hanesian, Education Psychology: A cognitive View, 1968.
[79] D.P. Ausubel, In Defense of Advance Organizers: A Reply to the Critics*, *Rev. Educ. Res. Rev. Educ. Res. Spring*. **48** (1978) 251–257. doi:10.3102/00346543048002251.
[80] C. Bizer, T. Heath, and T. Berners-Lee, Linked data-the story so far, *Int. J. Semant. Web Inf. Syst.* (2009). doi:10.4018/jswis.2009081901.
[81] M. Al-Tawil, V. Dimitrova, D. Thakker, and B. Bennett, Identifying knowledge anchors in a data graph, in: HT 2016 - Proc. 27th ACM Conf. Hypertext Soc. Media, 2016. doi:10.1145/2914586.2914637.
[82] R. Wille, RESTRUCTURING LATTICE THEORY: AN APPROACH BASED ON HIERARCHIES OF CONCEPTS, in: S. Ferré, and S. Rudolph (Eds.), Form. Concept Anal. 7th Int. Conf. ICFCA 2009 Darmstadt, Ger. May 21-24, 2009 Proc., Springer Berlin Heidelberg, Berlin, Heidelberg, 2009: pp. 314–339. doi:10.1007/978-3-642-01815-2_23.
[83] G. V Jones, Identifying Basic Categories, *Psychol. Bull.* **94** (1983) 423–428. doi:10.1037//0033-2909.94.3.423.
[84] M.A. Corter, James E.; Gluck, Explaining Basic Categories: Feature Predictability and Information, (1992).
[85] C.F. Palmer, R.K. Jones, B.L. Hennessy, M.G. Unze, and A.D. Pick, How Is a Trumpet Known? The "Basic Object Level" Concept and Perception of Musical Instruments, *Am. J. Psychol.* **102** (1989).
[86] W.N.F. Henry Kucera, Computational Analysis of Present-Day American English, *Am. Doc.* (1968).
[87] M. Al-Tawil, Intelligent Support for Exploration of Data Graphs, University of Leeds. Submitted (defence in December), 2017.
[88] Z. Wu, and M. Palmer, Verbs semantics and lexical selection, *Proc. 32nd Annu. Meet. Assoc. Comput. Linguist.* -. (1994) 133–138. doi:10.3115/981732.981751.
[89] B. Kules, R. Capra, M. Banta, and T. Sierra, What Do Exploratory Searchers Look at in a Faceted Search Interface?, *JCDL '09 Proc. 9th ACM/IEEE-CS Jt. Conf. Digit. Libr.* . (2009) 313–322. doi:10.1145/1555400.1555452.
[90] T. Nunes, and D. Schwabe, Frameworks of Information Exploration - Towards the Evaluation of Exploration Systems Conceptual View of an Exploration Framework, in: IESD@ISWC, 2016.
[91] J.W. Tanaka, and M. Taylor, Object categories and expertise: Is the basic-level in the eye of the beholder?, *Cogn. Psychol.* **23** (1991) 457–482. doi:10.1016/0010-0285(91)90016-H.
[92] D.R. Krathwohl, A Revision of Bloom's Taxonomy : An Overview, *Theory Pract.* **41** (2002).
[93] L. Freund, S. Dodson, and R. Kopak, On measuring learning in search: A position paper, in: Search as Learn. Work., 2016: pp. 1–2.
[94] S.C. Carr, and B. Thompson, The effects of prior knowledge and schema activation strategies on the inferential reading comprehension of children with and without learning disabilities, *Learn. Disabil. Q.* **19** (1996) 48–61. doi:10.2307/1511053.
[95] S.G. Hart, and L.E. Staveland, Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research, *Adv. Psychol.* **52** (1988) 139–183. doi:10.1016/S0166-4115(08)62386-9.
[96] Sandra G. HartLowell E. Staveland, Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research, *Adv. Psychol.* **52** (1988) 139–183.
[97] M. Al-Tawil, V. Dimitrova, D. Thakker, and Alexandra Poulovassilis, Evaluating Knowledge Anchors in Data Graphs against Basic Level Objects, in: ICWE'17 Int. Conf. Web Eng., 2017.
[98] A.I. Schein, A. Popescul, L.H. Ungar, and D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval, SIGIR '02, 2002: p. 253. doi:10.1145/564376.564421.
[99] M. Al-Tawil, V. Dimitrova, and D. Thakker, Using Basic Level Concepts in a Linked Data Graph to Detect User's Domain Familiarity, in: UMAP, Dublin, Ireland, 2015.
[100] G.A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information, *Psychol. Rev.* **63** (1956).