Cluster Comput
DOI 10.1007/s10586-017-0908-2

An architecture for the autonomic curation of crowdsourced knowledge

Alina Patelli[1] · **Peter R. Lewis**[1] · **Aniko Ekart**[1] · **Hai Wang**[1] · **Ian Nabney**[1] · **David Bennett**[2] · **Ralph Lucas**[3] · **Alex Cole**[3]

Received: 17 March 2017 / Accepted: 3 May 2017

Abstract Human knowledge curators are intrinsically better than their digital counterparts at providing relevant answers to queries. That is mainly due to the fact that an experienced biological brain will account for relevant community expertise as well as exploit the underlying connections between knowledge pieces when offering suggestions pertinent to a specific question, whereas most automated database managers will not. We address this problem by proposing an architecture for the autonomic curation of crowdsourced knowledge, that is underpinned by semantic technologies. The architecture is instantiated in the career data domain, thus yielding Aviator, a collaborative platform capable of producing complete, intuitive and relevant answers to career related queries, in a time effective manner. In addition to providing numeric and use case based evidence to support these research claims, this extended work also contains a detailed architectural analysis of Aviator to outline its suitability for automatically curating knowledge to a high standard of quality.

Keywords Knowledge curation · Semantic technologies · Ontologies · Autonomic computing

✉ Alina Patelli
a.patelli2@aston.ac.uk

Peter R. Lewis
p.lewis@aston.ac.uk

Aniko Ekart
a.ekart@aston.ac.uk

Hai Wang
h.wang10@aston.ac.uk

Ian Nabney
i.t.nabney@aston.ac.uk

David Bennett
david@codevate.com

Ralph Lucas
ralph.lucas@goodcareersguide.co.uk

Alex Cole
alex.cole@goodcareersguide.co.uk

[1] Aston University, Birmingham, UK

[2] Codevate, Birmingham, UK

[3] The Good Careers Guide, London, UK

1 Introduction

Decision making in the digital world is supported by effective knowledge processing. Given the size of the available digital data repositories, manual curation is fast becoming unfeasible. Automated query answering platforms (leveraging data from museum records [45], computerised tools for symptom based medical diagnosis inference [32], archaeological database processing [37], etc.) represent an attractive solution, however, several important issues remain unaddressed:

- The connections between different data entries are rarely and insufficiently exploited, therefore the results presented in answer to user queries lack insight and are often incomplete.
- The format that query results are presented in (commonly, lists of entries that syntactically match the search keywords) is counter-intuitive and unable to provide a coherent view of the relevant sub-field of the knowledge base.
- The provided results are rarely filtered based on the user's profile and interests.
- The user has to address the problems above "manually" by explicitly searching for additional results (maybe by employing several query answering tools and collating

Published online: 09 June 2017

their respective output), researching data connections and matching them against personal interests, etc.—all time consuming operations requiring intense effort.

We analyse these open problems in the career knowledge management domain, where the available data is abundant, heterogeneous, decentralised and dynamic. Yet, the workforce is expected to effectively analyse it in order to make informed decisions about the most suitable career path. For this reason, we believe the career domain offers a representative case study for investigating the proposed research question, namely how to design an automated knowledge curation platform capable of addressing all previously identified issues.

The proposed solution is Aviator, a career knowledge management system available on the GCG (Good Careers Guide) platform that stores, maintains and exposes the connections between career fields, displays query results in the form of an intuitively rendered graph (as opposed to a list), compares available knowledge against expressed user preferences and performs all these tasks automatically, thus saving a significant amount of the users' time.

Our initial work on Aviator [36] is extended here with a detailed qualitative analysis of Aviator's architecture—carried out according to the architectural tradeoff analysis method (ATAM). The suitability of Aviator in the career domain notwithstanding, the proposed architecture is fit for deployment in the general context of knowledge curation, as the ATAM outcomes reveal.

The following section presents the motivation for this research and more fully describes the problem that we address. Section 3 focuses on the career knowledge domain as a representative instance of the autonomic curation context. After a brief description of Aviator's hybrid architecture (Sect. 4), the paper analyses the way that the proposed platform implements the autonomic metaphor (Sect. 5). Evidence to support all research claims is provided in Sect. 6, whereas Sect. 7 contains the ATAM analysis. The final sections present an overview of related work and the paper's conclusions.

2 Motivation

Great strides have been made in recent decades to digitise information [3,12,14,16,35], as paper-based systems have been replaced by databases available over the web. In legacy paper-based systems, the role of the curator[1] was key. For example, when presented with a university student's query about "modern art", most librarians would be able to provide all the books on the official reading list. However, experienced librarians would also recommend less known yet relevant resources (websites, articles, critics' reviews) found useful by other library members on a similar academic quest. It is usually the insight provided by this sort of material that turns a good university essay into an excellent one. To provide an example from a safety-critical domain, let us think of medical staff as curators of knowledge. Decisions about patient treatment are based on the physician's core specialist knowledge about human anatomy as well as on specific case studies, recent research and other clinicians' experience in similar or more loosely related domains. It is often the connections between all those sources of knowledge that enable medical professionals to formulate an accurate diagnosis.

Given the ever increasing volume of information across all fields, the pool of resources the human curator should have expert knowledge of has become intractable. The IT community's solution to this issue was to transfer all available data from a paper support to a digital one. Ideally, the entirety of the human curator's knowledge should be captured by a (library, medical, etc.) database, whereas the curation role itself would be taken over by the database manager. Realistically, that aim was achieved only to a certain extent: while the core data (library cards, patient charts, known symptoms of medical conditions, etc.) was successfully ported from hard copy versions to databases, the *experience* of human curators, namely the *connections* they were able to make between different types of knowledge, was lost along with the sense of (library, medical, etc.) *community* that used to factor into the curator's decision making process. As a result, running a query for "modern art" in a digital database will no longer return the additional resources that do not exactly match the search keyword but that the human librarian had knowledge of. Similarly, a diagnosis based only on the results returned by a medical symptoms' database will not account for specific yet relevant cases that human doctors would know of and be able to interpret.

One way to address this became available with the dawn of Web 2.0 [34], a reinvention of the classic World Wide Web, where online content is curated by non-expert users. This is done by annotating web resources with *tags*, usually as simple as words, that concisely capture one aspect of the online content. For instance, a digital print of a Monet painting could be tagged with "water lily" to describe its theme and "blue" to refer to the predominant colour. This approach to online content management proved very attractive, with websites such as YouTube, Delicious, Flickr [10] and Pinterest [17] gaining increased popularity. The immediate advantage is that separate resources are connected via user tags, thus reinstating a sense of *community*, on the one hand, as well as allowing for better, more powerful search algorithms, on the other hand (if those additional library resources existed

[1] A content specialist charged with an institution's collections and involved with the interpretation of heritage material—retrieved from https://en.wikipedia.org/wiki/Curator.

on a Web 2.0 website and were tagged with "modern art", they would be included in the query results alongside the traditional matches).

However, an important problem remains. Left unchecked, that is, under the exclusive control of the individual user, tags may quickly become ambiguous (a different user may tag the same Monet painting with "pond flowers"), conflicting (a given viewer may be of the opinion that the predominant colour in the print is "green", not "blue") or incorrect (the print may be wrongfully tagged with "Manet" instead of "Monet"). Thus, the added value brought by community curation turns against itself and sabotages the powerful search algorithms it was meant to support. One possible solution lies with the Semantic Web [4], another iteration in the World Wide Web's transformation, where user tags are regulated by an ontology [6,31,40]. An ontology stores concepts and the properties connecting them in the form of a graph, expressed in the light logic formalism of RDF (Resource Description Framework) [11]. The lead advantage provided by an ontology in the online curation context is disambiguation: every concept is represented alongside its synonyms (maintained by an expert, by the larger community, or, ideally, by both) that can be used by search algorithms to identify equivalent tags and eliminate conflicting ones.

Ontologies offer intrinsic support for some of the challenges identified in the introduction (they store synonyms, therefore are capable of running "richer" queries, with a better yield and they are structured as graphs—with concepts in the role of nodes and properties acting as edges—knowledge models that are intuitive and easy to explore in order to get a comprehensive perspective of the relevant sub-field). However, taking user preferences into consideration in order to produce relevant query results implies some supplementary logic that ontologies do not provide native support for. Also, running complete queries, allowing graph exploration and filtering results based on user profile are all tasks that need to be executed automatically, which is again beyond the core capabilities of ontologies as standalone platforms.

3 The career management scenario

In this paper, we tackle the above challenges in the domain of career management platforms, which are a typical example of a knowledge base in transition from paper to the digital world. Further, effective career management platforms are crucial tools in providing support for informed decision making for the entire workforce.

In its current form, the online career space comprises knowledge from three sources:

- **experts** (National Careers Service, relevant Wikipedia pages, etc.) providing general information about professional fields and the way they connect to each other, for instance, the fact that "chemistry" is a sub-field of "science"
- **providers** of either education (universities publishing academic requirements for pursuing a given career, HESA[2] maintaining the latest JACS[3] list) or jobs (company websites offering specific career / role description, job adverts published via third party websites, such as http://indeed.co.uk)
- **explorers** of online, career relevant content in search of a new job or a better understanding of their professional prospects and assigning tags or writing reviews in the process.

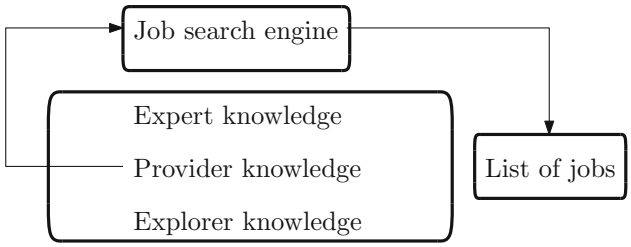


Fig. 1 The career space managed by job search engines—expert and explorer knowledge is ignored

The only form of automatic career knowledge management available for explorers is provided by job search engines (e.g., http://indeed.co.uk, http://jobs.ac.uk), as shown in Fig. 1. These take in one or more keywords and produce a list of job adverts based on syntactically matching the provided keywords against the text description of the jobs. Besides the semantic incompleteness of the results (relevant jobs may be omitted from the list if published under a synonym of the search keyword that the explorer is unaware of), such search engines disregard the first and third knowledge sources altogether. The *connections* between career concepts as well as the explorer *community* output (in the form of tags and reviews) are thus obscured. Consequently, explorers take the curator's role upon themselves and sift through HESA content to match their academic credentials against job requirements, read generalist web pages with broad scope information about each role in the result list and consult other explorers' reviews and comments in order to make an informed decision about applying for a given job or not.

We introduced Aviator[4] [36], a career management platform available on the Good Careers Guide platform that allows explorers to tag career resources with concepts from

[2] Higher Education Statistics Agency—https://www.hesa.ac.uk/.

[3] Joint Academic Coding System—https://www.hesa.ac.uk/component/content/article?id=1787.

[4] https://gcg-test.codevate.com—log in with user name "johnsmith" password "gcgtesting".

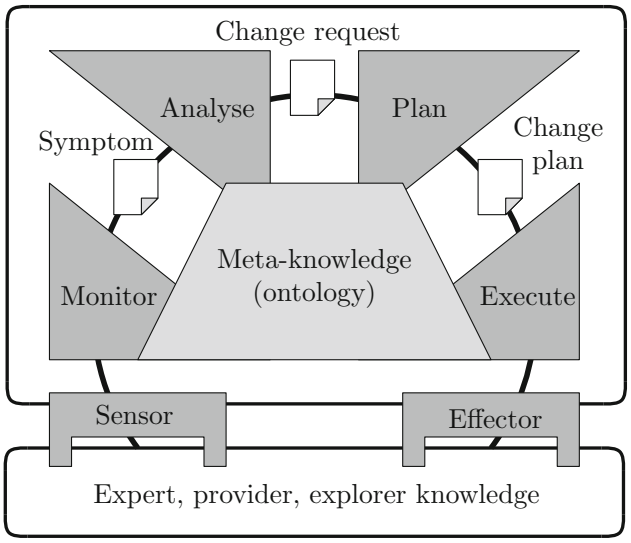


Fig. 2 The career space (*lower block*) managed by an autonomic manager (*upper block*). Adapted from [21]

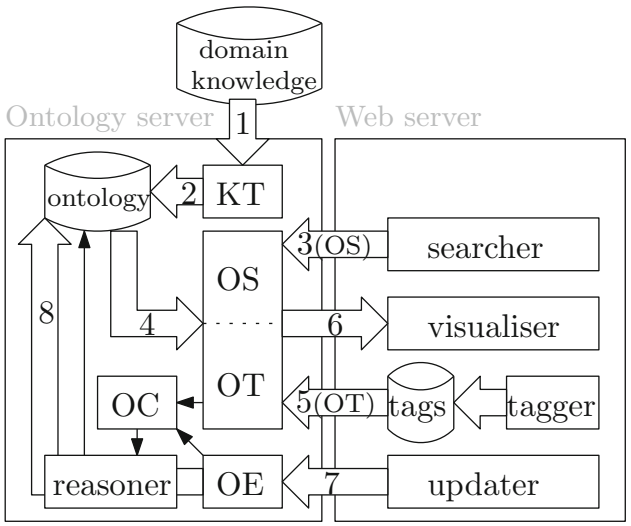


Fig. 3 Aviator architecture: *line arrows* represent "uses" relationships between components, *block arrows* illustrate the data flow through the system; *KT* knowledge translator, *OS* ontology segmenter, *OT* ontology tailor, *OE* ontology editor, *OC* ontology classifier

an ontology and benefit from each other's expertise. Aviator addresses the previously identified issues by:

- offering **completeness** in the sense that all synonyms of a career concept known to the system will be considered when compiling the associated list of jobs
- providing **perspective** by displaying the career concepts relevant to the user query as well as their connections (the latter are unavailable in the classic list format that job search results are displayed in)
- enhancing **relevance** via collecting all the tags that a registered user annotated online resources with and using them to generate a personal ontology—this can be compared against the ontology of the ideal candidate for a given role, thus allowing jobs that do not match the user's career profile to be filtered out
- saving **time** gained by having queries answered in a complete and automatic fashion, rather than manually curating the relevant knowledge.

4 Architecture description

Looking at Aviator's architecture from a high-level standpoint, there is a parallel to be drawn to IBM's standard autonomic element model [21,26]. Specifically, the role of the curator is fulfilled by an autonomic manager (the upper block in Fig. 2), where the knowledge component of the underpinning control loop (referred to as MAPE-K, namely monitor analyse plan execute—knowledge) is an ontology.

The *monitor* collects information from **providers** (new jobs posted on http://indeed.co.uk) and **explorers** (tags, reviews, ontology edits). The *analyse* module verifies the consistency of the underlying knowledge base, accepting/rejecting edits accordingly, and maintains the list of tags used by every registered system user. The *plan* component runs either a simple query (to retrieve the segment of career knowledge that the user is interested in exploring) or a compound one (essentially, a separate query for each tag) to compile a personal ontology. The results of the query are displayed in the *execute* phase as a graph in the system's visualiser. The *knowledge* informing the operation of the MAPE loop is represented in the form of an ontology, initially extracted from a legacy document containing **expert** knowledge about career concepts, their properties (synonyms and connections) and the relevant JACS codes. The ontology is maintained via user edits and displayed (in segments) in response to user queries.

At a more finely grained architectural level, the two main Aviator components (illustrated in Fig. 3) are the *web server* hosting the user interface and the *ontology server* providing a feature rich semantic platform capable of running user requested services (e.g., incremental graph exploration, graph editing, personal ontology generation). The ontology server performs the functions of the autonomic manager (upper block in Fig. 2): monitoring is realised by importing expert and provider domain knowledge (flow arrow 1 in Fig. 3) as well as explorer input (flow paths 7 and 8); analysis, planning and execution are implemented by the ontology segmenter, tailor and editor (the exact mapping is

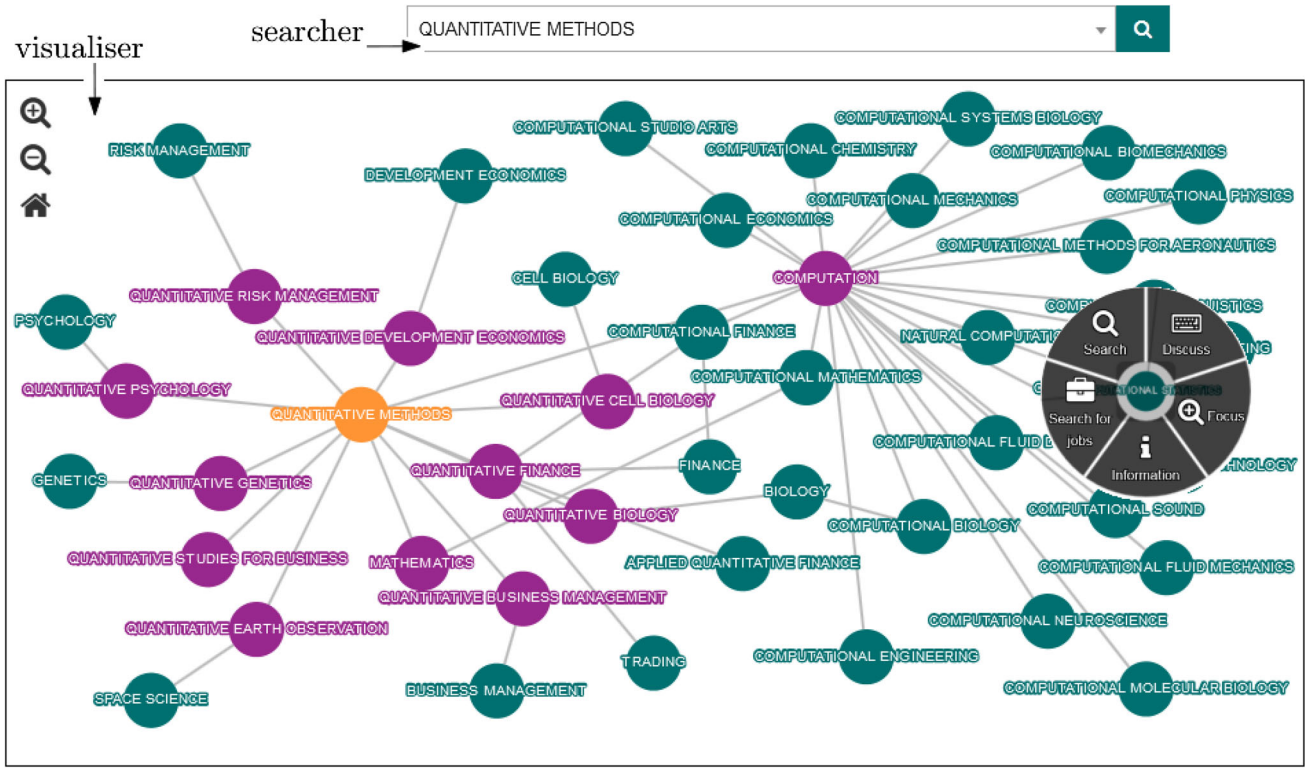


Fig. 4 Aviator: *searcher* and *visualiser* components

presented in Sect. 5); the knowledge base is the ontology itself.

4.1 The web server

The *searcher* takes a keyword from the user and matches it against ontology concepts. The ontology nodes related (via a maximum of two links) to the matching concept are displayed in the *visualiser* (the visualisation plug-in used by Aviator is Cytoscape[5]). To illustrate the process, Fig. 4 shows the result of the search for keywords "quantitative methods".

The website *tagger* (Fig. 5) displays as a menu to the left of the page being annotated and allows users to assign tags (i.e., concepts from the ontology) to its content. If the page has been previously reviewed by other users, that information is available in the menu as well. Every ontology concept used as a tag by a specific user is stored in the *tags* collection.

The *updater* (available only to administrators) allows the editing of the ontology by adding/deleting concepts and their connections through a dedicated graphical interface. Figure 6 illustrates the process of creating a new link (asserting a new ontology property) connecting "biological computing" (the *Source*) to "applied biological sciences" (the *Destination*). The system displays a list of existing ontology concepts currently related to the source and destination (e.g., "biological computing" currently has two parents, "biology" and "applied computing"), which is meant to inform the user's decision with respect to the most appropriate type of link to assert. In the example in Fig. 6, "biological computing" is made the child of "applied biological sciences" (the other two options are parent or sibling).

4.2 The ontology server

The *ontology* is expressed in OWL[6] and extracted from a data repository (marked *domain knowledge* in Fig. 3) provided by a domain expert. Besides career related concepts, the ontology also stores relevant JACS subject identifiers, thus making the Aviator ontology compatible with HESA and UCAS standards for UK higher education. The relationships between career nodes are expressed via three semantic properties, namely `hasParent`, `hasSibling` and `hasSynonym`. The ontology is created by the knowledge translator from expert provided career data (flow arrow 2 in Fig. 3) and is modified by the ontology editor, which is in charge of implementing user updates (flow arrow 8 in Fig. 3). Ontology content is fed into the ontology segmenter and the ontol-

[5] http://www.cytoscape.org/.

[6] http://www.w3.org/2001/sw/wiki/OWL.

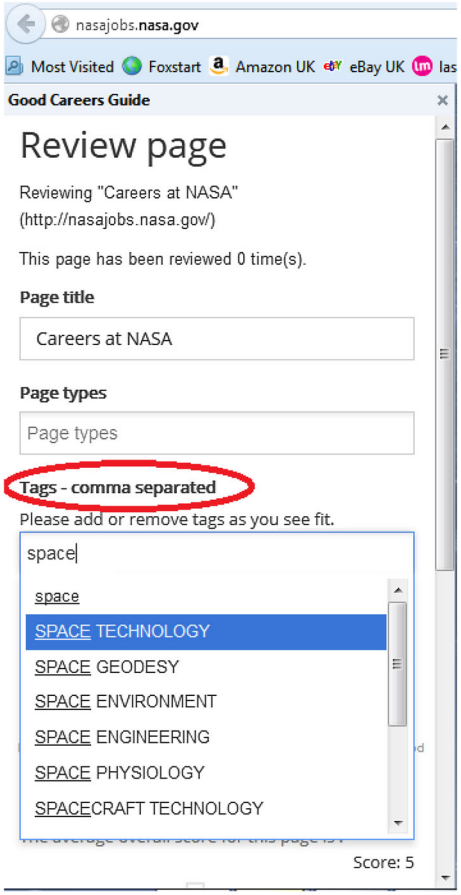


Fig. 5 Aviator: the *tagger* component used to annotate http://nasajobs.nasa.gov

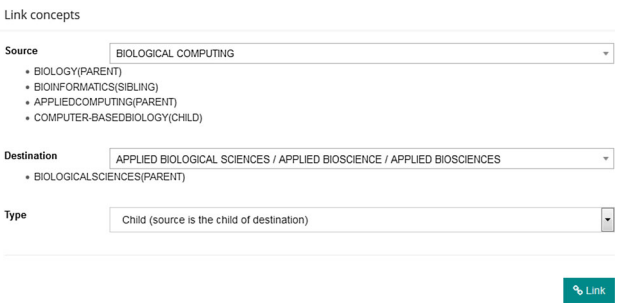


Fig. 6 Aviator: the *updater* component

ogy tailor, responsible with creating custom graph views in response to user requests (flow arrow 4).

The *knowledge translator* (*KT* in Fig. 3) populates the ontology by transforming domain knowledge into semantic contents. This process corresponds to data flow stages 1 and 2 in Fig. 3.

The *ontology segmenter* (*OS*) and the *ontology tailor* (*OT*) are represented in the same block in Fig. 3, as they share input 4 (feeding from the ontology) and output 6 (displaying the generated results in the visualiser). Input 3, namely the keyword used in the search, is specific to the OS only. Input 5, that is, the collection of tags (ontology concepts) that the current user annotated career webpages with, feeds exclusively into OT. In terms of actual operation, OS matches a search keyword against an existing ontology concept c (input 3) and runs a DL query over the ontology (input 4) to extract a set of nodes related (over a maximum of two connections) to c via `hasParent` or `hasSibling` properties. The resulting ontology segment is fed into the visualiser (output 6) where it is displayed as a graph. The *ontology tailor* (*OT*) performs the same operation as the OS, only in batch mode, once for every element in *tags* (input 5). Each query will produce a graph, their ensemble forming the current user's personal ontology (output 6). These are useful for job seekers as they represent visual descriptions of their professional interests, in other words, a history of their job related web browsing.

The *ontology editor* (*OE*) receives the modification suggested by the user (e.g., a concept/link addition/deletion) through the updater (input 7) and asserts it in the ontology (output 8). There are three types of edits currently available through the ontology administration interface: turning a node into a synonym and vice versa, adding a new concept (the parents, siblings, children and synonyms of the added node need to be specified as well) and adding/deleting a link. The addition of a new link via the updater (making "biological computing" a child of "applied biological sciences") is illustrated in Fig. 6. Given the sensitive nature of the edit operation (that allows end users to modify the knowledge base), the updater is currently only available to administrators.

The *reasoner* is meant to maintain the logical consistency of career related knowledge as well as infer new knowledge to support the ontology search process. The *ontology classifier* (*OC*) is the component in charge of deploying the reasoner whenever necessary (e.g., before committing changes to the ontology, to ensure logical consistency is maintained).

5 Autonomic curation

This section explains how autonomic curation of crowdsourced knowledge is implemented in the context of the above architecture.

While the system is running, the components in Fig. 3 interact in a way that can be described as a MAPE-K loop (Fig. 2). Thus, the Aviator platform may be viewed as an autonomic system [26], where the careers' knowledge space (authored by experts, providers and explorers) is the managed resource (curated knowledge) and the remaining components of the ontology server make up the autonomic manager (curator). The goal of the system is to offer completeness of

search results, provide a perspective of the wider careers field, enhance the relevance (with respect to personal professional interests) of career related searches and save the user a significant amount of time by automating all above tasks (see Sect. 3). These four sub-goals represent a mix of qualitative and quantitative benefits and it makes little sense to aggregate them in a conventional objective function. However, an initial evaluation of these sub-goals is provided in Sect. 6.

The components of the autonomic manager are described in the following.

5.1 Monitor

The Aviator system monitors:

- S1 provider knowledge (new jobs posted on http://indeed.co.uk)
- S2 explorer knowledge (new tags utilised by registered users to annotate, via the tagger in Fig. 5, online career resources)
- S3 expert knowledge (expressed by editing the ontology, namely adding, deleting or redefining ontology concepts and properties via the updater in Fig. 6).

The web server provides the "software sensors" to capture changes in the three knowledge sub-spaces and pass them to the appropriate ontology server components. The monitoring behaviour of the Aviator system is described by the following pseudo-code.

```
1  monitor(Sensor[] sensors)
2
3  while(true)
4    foreach s in sensors do
5      if s.isActive() then
6        analyse(s);
7      end if
8    end for
9  end while
```

List `sensors` contains S1 through S3, method `isActive()` returns whether sensor `s` has detected a change and `analyse()` is the method that represents the analyse phase.

5.2 Analyse

Analysis mostly consists in discriminating between the several types of monitored requests (via the `getType()` method), translating the sensor data (retrieved by `getOutput()`) to the right format and selecting the appropriate plan. The second `analyse()` input represents the author of the change detected by sensor `s`. The pseudo-code describing the analysis phase is presented below.

```
1  analyse(Sensor s, Author a)
2
3  switch(s.getType())
4    case S1:
5      keywords = parse(s.getOutput());
6      for each k in keywords do
7        c = getConcept(onto)
8        assign(s.getOutput(), c);
9      end for
10   case S2:
11     clearVisualiser();
12     tags = s.getOutput();
13     plan1(tags);
14   case S3:
15     if author.isAdmin() then
16       update = s.getOutput();
17       (isConsistent, tempOnto) = plan2(update);
18       if isConsistent == true then
19         onto = tempOnto;
20       end if
21     end if
22 end switch
```

In the case of S1, the sensor data returned by method `getOutput()` is the new job post. Method `parse()` extracts the post's `keywords`, which are then matched against ontology concepts (line 7). The new job post is included in the list associated to each previously identified concept `c` (line 8). Method `clearVisualiser()` resets the graph display (Fig. 4), whilst `plan1()` and `plan2()` refer to the selected plans. The administrator privileges of the detected change's author are either confirmed or invalidated by method `isAdmin()`.

5.3 Plan and execute

In order to generate a user's personal ontology (the responsibility of OT in Fig. 3), it is necessary to retrieve all concepts used as `tags` throughout the user's web exploration history and run a semantic query for each of them. The latter task is performed by the OS (see Fig. 3) by delegating to the reasoner. The query output is the matching concept's vicinity (`view` in the `plan1` pseudo-code) or null (in case the keyword did not match a concept). Method `display()` compounds the views generated for each tag and displays them in the visualiser. The same plan is used to explore the general career ontology (the centrepiece in Fig. 2), in which case the `tags` input is replaced by the keyword typed in the searcher and the `for` loop on line 3 becomes unnecessary as lines 4 and 5 need only be executed once.

```
1  plan1(Tag[] tags)
2
3  for each tag in tags do
4    view = OS(tag);
5    display(view);
6  end for
```

Edits formulated by users are performed on a temporary copy of the ontology which is afterwards submitted to the reasoner for consistency checking. The reasoner output will then be analysed (case S3 in `analyse()`) and acted upon

by either committing the changes to the public ontology or dismissing them altogether. The associated plan is:

```
1  plan2(Change c) returns boolean isConsistent,
2                          Ontology tempOnto
3
4  (isConsistent, tempOnto) = runReasoner(change);
5  return (isConsistent, tempOnto);
```

Line 4 above describes the function of the OE. The reasoner will return the updated ontology along with a flag indicating whether logical consistency is met or not.

The "software effector" executes the steps of `plan2` on the ontology and those of `plan1` on the front-end display, namely the visualiser in Fig. 3 (the latter operation is supported by the Cytoscape plugin). Specifically, the effector executes one of two management actions: commits changes to the ontology after a reasoner-approved edit or displays a sub-view (either single query output or personal ontology) of the graph in the visualiser.

6 Case study evaluation

The claimed benefits of autonomic curation of crowdsourced career knowledge are completeness, perspective, relevance and time. This section evaluates Aviator's capacity of practically realising these four benefits. A brief analysis of the platform's realtime operation is also provided.

6.1 Completeness

Let us assume that a user is interested in getting a job in *advertising*. The results list provided by http://indeed.co.uk for that keyword used on its own[7] contains 864 job adverts. However, the ontology features several synonyms for the concept "advertising" (Fig. 8), which, when considered together (via "Advanced search", in the textbox labeled "With at least one of these words"), form a query that yields a result list with 882 entries. To get access to these 18 extra jobs, the user would need to manually compile a list of all "advertising" synonyms, a task successfully automated by Aviator. The difference is even more striking if, by chance, the user searches for "creative director", which produces 21 results (thus, a negative difference of 861 jobs relative to the Aviator query).

Since the lists of jobs found for each of the "advertising" synonyms are overlapping, the complete query performed by Aviator merely broadens the result set (the jobs at the intersection of the results' lists would be identified by any of the individual queries). However, the added benefit brought by Aviator queries becomes more evident in cases where there is no overlap between the lists obtained for each career synonym. For instance, "business intelligence" and "business information management", taken as two individual queries, yield 252 and, respectively, 3 results. When aggregated in a single query (in the Aviator ontology, they are synonyms), the result set contains all 255 results, showing that the lists generated for the individual queries are completely distinct. In such a situation, without Aviator support, a user who is unaware of the synonymy relationship would be deprived of the entirety of jobs associated to either one field or the other. Thus, Aviator replicates the domain expertise of a curator, in automatically referring the user to other jobs of interest, even though they were outside of the user's specific search.

[7] All http://indeed.co.uk search results in this paper refer to searches conducted in May 2016 in the Birmingham area with a radius of 25 miles.

6.2 Perspective

Entering a keyword in the Aviator searcher will produce a graph containing relevant ontology nodes *as well as their connections* (this can be tested by navigating to https://gcg-test.codevate.com/explore and exploring the graph returned for any preferred ontology concept). Displaying the output of a query as a graph provides the user with an overarching perspective of the field of interest, which is not available (or, at best, severely obscured) when presenting query results in the form of a list (such as relevant job adverts are formatted by http://indeed.co.uk, for instance). Moreover, from a human computer interaction perspective, research shows that displaying knowledge as graphs is more informative than lists (indented trees) [15,24]. Another, indirect, advantage of displaying a career connectivity map rather than a list is the possibility of uncovering new, potentially relevant careers that the user may not have considered otherwise. For instance, searching for "biophysics" produces a graph featuring the expected connections ("biology" and "physics" are parents of the search topic) as well as unexpected ones ("astrobiology" is also a child of "biology" and "physics" and may be of interest, even if merely borderline, to a person with expertise in "biophysics"). This new connection is another example of something that might frequently have been pointed out by an expert curator, but would have been difficult to spot in a list of job adverts returned from a keyword search.

6.3 Relevance

A personal ontology comprises all Aviator concepts used as tags by a given registered system user. Those concepts, along with their one-step neighbours, are connected to a central node (labeled "Me" in Fig. 7) and displayed in the visualiser (https://gcg-test.codevate.com/explore/personal). Besides acting as a personal career profile (reflecting users' professional interests throughout their use of Aviator), a per-

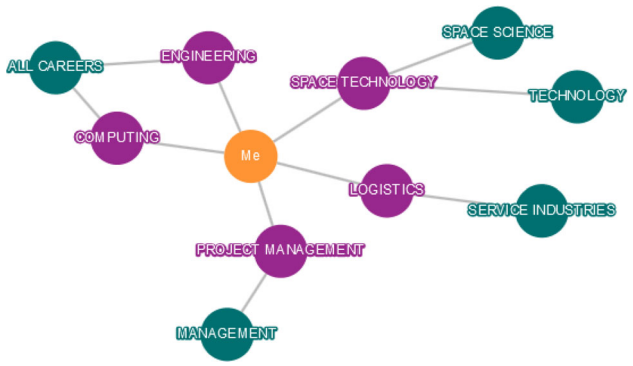


Fig. 7 Aviator: a personal ontology example

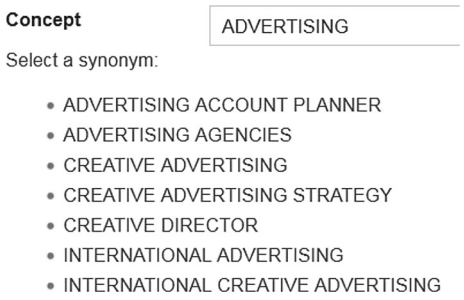


Fig. 8 Synonyms for "advertising" in the Aviator ontology

sonal ontology can serve as a benchmark when searching for jobs.

Specifically, let us assume that a fictional organisation, CompX, registers an Aviator account and publishes the personal ontology for the ideal candidate they would like to hire to fill a given role. In this example, CompX's personal ontology contains all the nodes in Fig. 7 (where "Me" is also replaced with "CompX"), apart from the "project management" branch. Let us also assume that John, a young engineer and owner of the personal ontology in Fig. 7, wants to find out if CompX's job offer is a good fit for him. Upon seeing the degree of overlap between his personal ontology and the one published by CompX, John can make an informed decision with respect to either applying for the job or not. Specifically, John may decide to keep looking for roles that also require project management skills or "sacrifice" his interest in that field and apply for the CompX job. By performing this comparison against a wide range of ideal candidate ontologies, users are supported in applying only for those jobs that are relevant to their professional interests.

6.4 Time

Using Aviator to manage career data saves the end user's time mainly in two ways. Firstly, by storing a list of synonyms for each ontology concept, Aviator supports broader scope queries. To achieve a similar result, the explorer would have to manually compile the synonym list from various sources, a time consuming activity thus rendered unnecessary. Secondly, comparing personal ontologies against those describing ideal candidates for available roles protects the end user from having to explicitly investigate the overlap between a given job description and personal professional interests.

Table 1 Ontology operations duration

Operation	Runtime (s)
(a) One-off operations	
KT ($1 \rightarrow 2$)	2
KT ($2 \rightarrow 1$)	3.6
OS (4)	0.9
Node links	**Query runtime (ms)**
(b) Queries	
< 50	132
50–100	532
>100	1231

6.5 Scalability

In support of Aviator's scalability to knowledge bases of different sizes, we provide numerical evidence of the platform's realtime performance. The main semantic operations and their execution times are listed in Table 1. Table 1(a) shows the duration of all operations that get run only once per user session: knowledge translation, both from xlsx to RDF/OWL, KT($1 \rightarrow 2$), and vice-versa, KT($2 \rightarrow 1$), and ontology loading prior to running a semantic query, OS(4). The associated execution times are measured on a DigitalOcean server: 4 CPUs @ 8GB RAM for the ontology server and 2 CPUs @ 4GB RAM for the web server. Table 1(b) lists average query execution times measured for three types of nodes: loosely connected (with less than 50 first and second order children, parents and siblings), well connected (between 50 and 100 related concepts)—this is the category 75% of ontology nodes fall under—and highly connected (with over 100 neighbours). The number of node neighbours is the only parameter considered since it has the most significant impact of the computational cost of semantic queries. The values in column two represent the average execution times for 100 ontology nodes from each of the three categories. As expected, query runtime increases as node connectivity goes up, however, for 75% of all possible queries, the execution time is under one second. The most computationally expensive operation performed by Aviator, ontology classification, employs the FaCT++ reasoner [47] and takes, on average, 91s.

7 Aviator architecture evaluation

In order to evaluate the suitability of the proposed platform for autonomic knowledge curation, in general, we carried out an architecture analysis. The tool used for this purpose, namely, the architecture trade-off analysis method (ATAM) [25] is the industry standard approach for assessing software applications' design against their intended purpose. It provides a well structured procedure to evaluate an architecture's fitness taking into account a set of non-dominant quality attributes (in the sense that improving one attribute will implicitly worsen another). ATAM has been successfully used for analysing software architectures in cloud computing [30], with a special focus on the security aspect [13], for investigating the design of service-oriented systems for serious games [8] as well as a starting point for bespoke approaches considering the impact of uncertainty on software requirements and architectures [29], the importance of enterprise information systems availability [33] or the sustainability of software architectures [50].

When applied to the Aviator domain, ATAM enabled the identification and management of the following factors.

7.1 Business drivers

The architecture should allow heterogeneous information *integration*, intuitive *navigation* of that information and *personalisation* of reports and search results.

7.2 Architectural plan

The proposed architecture is an ontology supported autonomic manager. It addresses the previously identified business drivers in that autonomic managers are specifically designed for *unsupervised*, *realtime* operation, ontologies store heterogeneous information uniformly (thus supporting *integration*) as semantic graphs (intuitively *navigable* structures) and reasoners support semantic querying, a powerful instrument for providing insightful, *personalised* results.

7.3 Architectural approaches

To assess the robustness of the plan, several alternatives were investigated with respect to the *architecture* of the autonomic system (identified as the main discrimination criteria in the literature [20,21,26]) as well as the *type* and *structure* of the knowledge layer (presented as significant decision factors within the ontology engineering process [38,42]).

With respect to the autonomic management system, the following options were considered:

- hierarchical vs flat autonomic system
- distributed (networked) vs local autonomic system
- off-the-shelf autonomic development platform (such as IBM's Autonomic Computing Toolkit [22]) vs bespoke implementation.

In what concerns the type of the knowledge layer, several options were analysed as alternatives to ontologies:

- relational database
- NoSQL database (object oriented or graph model)
- formal model (e.g., temporal logic).

Selecting the type of knowledge to effectively support the manager's decisions is not a trivial task [21,27]. Consequently, the structure of the knowledge layer was discussed in terms of the options below.

- inclusion vs exclusion of plans and policies from the main knowledge repository
- inclusion vs exclusion of system states in the knowledge repository
- multifaceted properties vs RDF triple representation—⟨subject, predicate, object⟩.

For an extensive analysis, one may consider architectural combinations across the three categories of approaches presented above (e.g., compare a hierarchical autonomic manager with a relational database serving as its knowledge layer against a distributed manager informed by a NoSQL database). However, based on the quality attributes preferred by the stakeholders (see Sect. 7.4 below), it was possible to eliminate one of the candidates for each item in the lists above without considering cross-category combinations.

7.4 Quality attributes and usage scenarios

The stakeholders, namely the technical architect, the product owner, the project manager, the lead developers and a focus user group, established a set of relevant quality attributes. They are described below in order of priority (low to high) and illustrated with practical scenarios.

7.4.1 Proactivity

Systems in charge of autonomic knowledge curation should make and implement decisions with minimum human intervention. Heterogeneous information should be collated, interpreted and displayed in the absence of human supervision.

Usage scenarios[8] Users are exploring and tagging career related resources, while their annotations are captured by

[8] All usage scenarios in this section refer to the career knowledge curation case study.

a sensor and integrated in the underlying knowledge base without interrupting their browsing. New job postings and relevant user reviews are captured automatically. Administrator edits operated on the ontology are accepted or rejected, algorithmically, given the credibility scores of their authors.

7.4.2 Adaptive transparency

In some application spaces (e.g., autonomic medical diagnosis or realtime navigation systems), the knowledge base should be hidden from the end user to minimise disruption. In others, as is the case with Aviator, allowing user access to the knowledge stored by the autonomic manager is highly beneficial, especially when understanding the underlying data is vital to the type of service provided.
Usage scenarios Career advice seekers benefit from having access to graph-organised data about various professions. This exposes the way career nodes relate to each other, to jobs, to relevant university courses, etc., thus providing users with a perspective of the field.

7.4.3 Realtime operation

The systems underpinned by the proposed architecture must respond to change without significant lags. Although some career knowledge base maintenance may be performed offline (for instance, complete consistency checks usually take place once a day), data must be collected as it is published on job sites or annotated by career resource explorers.
Usage scenarios. People doing online research about careers annotate a webpage while a new job is advertised on http://indeed.co.uk—both streams of data are captured by the appropriate sensors and included in the knowledge base as they become available.

7.4.4 Structural simplicity

No additional level of complexity (at hardware or software levels) should be necessary to allow the system to meet a satisfactory standard of quality. This is of paramount importance, because simplicity supports flexibility, making the architecture adaptable to various problem domains.
Usage scenarios Career information management (by the system) and exploration (by the user) does not require a complicated "translation" layer to turn the language of the user into one interpretable by the machine. Besides a web server configuration that is appropriate for the expected user load, the manager does not require additional hardware to run on.

7.5 Architectural approaches' analysis

The proposed architectural plan (Sect. 7.2) was analysed in various configurations (listed in Sect. 7.3), with respect to the identified quality attributes. The resulting tradeoffs (that is, the quality attributes improved by a certain architecture as opposed to the ones damaged by it) were evaluated and a decision was made either in favour or against each architectural approach.

With respect to the autonomic system's architecture:

- A hierarchical autonomic system (comprising several managers on different levels of abstraction, controlled by hierarchical superiors) would improve the coordination of tasks throughout the system, potentially enabling the processing of a wider range of events (new jobs being posted, user reviews being added to career web pages, etc.) and a more accurate response (increased proactivity). However, that would also damage realtime operation given the lags implied by the communication between managers on different layers of the hierarchy. Since the latter quality attribute takes priority over the former, a flat architecture was adopted.
- A distributed autonomic system would help process large volumes of information faster (beneficial in terms of proactivity and realtime operation) yet the additional hardware and software inherent to a networked design would compromise simplicity. Thus, the architecture will contain one autonomic manager (as opposed to several spread across a grid), yet, the underpinning algorithms will be modular (configured as services) and capable of running on a PC as well as on a web server.
- A third party autonomic development platform would most likely improve proactivity (according to the way professional frameworks, such as ACT [22] or ABLE [5], are advertised). At the same time, "one-size-fits-all" solutions tend to be heavyweight and difficult to configure, which damages the simplicity and adaptive transparency quality attributes (at the time of writing, ACT does not allow the exposure of the underlying knowledge base to the user). Consequently, the proposed architecture will be built from modular tools, that are flexible both in operation as well as in the way they can be assembled together.

In what concerns the type of the knowledge layer, relational databases provide no native support for hierarchical knowledge (crucial when modelling related careers) and do not facilitate learning (only explicitly asserted facts are considered to be true, under the closed world assumption [40]). Therefore, this sort of system would have a limited capability of making decisions without being prompted (proactivity would be low). NoSQL or a formal model prove more flexible, yet would require a bespoke inference engine to support learning (damages simplicity). Also, a supplementary translation layer would be necessary to allow non-specialist users to understand knowledge expressed in a mathematical/logical formalism. The final decision was in favour of ontologies,

as they are equipped with embedded inference engines (reasoners) and lend themselves well to intuitive visualisation techniques, by exploiting the graph-like structure of RDF.

The discussion around the structure of the knowledge layer revealed that:

- Storing plans and policies in the ontology would allow the system to reason on these two components, thus increasing the accuracy of responses to complex triggers from the managed resource (i.e., the online career knowledge space). At the same time, the additional reasoning complexity would damage realtime operation and simplicity. Hence, in the proposed architecture, the two components are stored separately from the main knowledge repository.
- The availability of system states in the knowledge base would improve the efficiency of autonomic tasks such as analysis and planning (thus increasing proactivity and improving realtime operation). Yet a complete state model of the managed resource is not always possible to extract, not to mention the ensuing increase in the size of the knowledge base (hence decreasing proactivity and damaging realtime operation). Since this design alternative seems to be both beneficial and detrimental relative to the same quality attributes, the final conclusion was to include a state model in the knowledge repository of the reference architecture, yet instantiate it only when needed in the context of a given application (e.g., modelling the states of the managed resource in a discrete, small sized domain, such as automatically configuring an industrial actuator, is straightforward, whereas the same task in the context of career knowledge management is not computationally feasible).
- The inclusion of multi-faceted properties yields the same discussion as in the case of system states. Including them would allow for a more accurate representation of the managed resource with benefits in terms of proactivity, and, at the same time, would add a layer of complexity to the ontology, damaging the same quality attribute. As previously, the final decision was to provide the reference architecture with a mechanism for multi-faceted property formulation, with an optional practical realisation.

7.6 Additional usage scenarios

A prototype implementing the proposed architecture in the career knowledge curation problem domain was piloted during several advisory group meetings, where Good Careers Guide employees and Aston University students experimented with the platform and gave feedback. The additional usage scenarios that were formulated with this opportunity prompted minor operational changes (e.g, while exploring the knowledge base in the form of a graph, the edges should not be labelled with the type of relationship they represent, as that would clutter the display). However, those were accommodated without any modifications to the architecture agreed upon in the previous ATAM step.

7.7 Discussion

The proposed architecture therefore meets the trade-offs preferred by the stakeholders. This fact notwithstanding, another note-worthy candidate was carefully considered during the ATAM analysis, namely an autonomic manager informed by a NoSQL database. More flexible than a relational database, this sort of knowledge layer would increase the response time of the platform as a whole without requiring any supplementary computational resources (thus meting both the realtime operation and structural simplicity criteria). Yet, NoSQL databases provide limited support for intrinsic learning [28] (lagging behind ontologies in that respect) and are not as easily translatable to an intuitive visual form such as a graph—thus damaging the proactivity and adaptive transparency attributes. However, these two are of lower priority than realtime operation and structural simplicity—the true reason behind rejecting the NoSQL knowledge layer was the type of learning it enabled. Specifically, the online career knowledge space is dynamic, subject to continuous transformation. In order to remain relevant, the proposed career data management platform requires a knowledge base capable of absorbing such changes with minimum computational costs. Out of the knowledge modelling alternatives available at the moment, ontologies are shown to have the most promising potential for organic growth, synchronous to the natural frequencies of the domain being modelled.

8 Related work

Autonomic curation of online knowledge has received limited attention from the research community. Contributions usually target specific applications, such as curating metadata associated to digital records with the purpose of cataloging those for long-term storage [2,43]. A similar idea to that underpinning Aviator is used to allow the community-led curation of artworks in a digital gallery [18], however, the curation process has to do with the users' artistic preference rather than semantic content. None of these contributions use an ontology to store the knowledge piece of the autonomic manager nor give any insight into the MAPE-K loop they employ.

On the other hand, the area of online career support has proven more popular. Several career support platforms make use of ontologies to store and maintain relevant knowledge. The Enterprise Ontology [48] stores the vocabulary for the business enterprise domain. Career ontologies in the ICT

field are either aimed at facilitating the access of school leavers to the ICT curriculum, jobs, skills, etc. [9] or focus on improving the ontology search process (complete with a metric for measuring the relevance of ontology concepts with respect to user keywords) [39]. Other approaches [1,19] build ontologies from secondary school student data (psychological test results and exam marks) as well as expert provided career data (only broad domains, such as literature, humanities, mathematics, are considered). By matching student data against career requirements, the systems will recommend the best fitting field of professional practice [1] or the necessary courses to take in order to meet a given degree's promotion criteria [19]. A career advice platform is used as a case study to illustrate knowledge maturing [51], namely the process of transforming highly conceptualised entities into formal, explicitly linked concepts. The platform suggests the inclusion of knowledge graph visualisation components and analyses the benefits of effective retrieval of relevant information, yet the discussion is exclusively carried out at a design level. Another approach [52] uses knowledge graphs to allow an easier understanding of mathematical concepts and mainly focuses on how to manually build the graph rather than extract it from a legacy repository.

We also analyse a body of work dealing with *knowledge graphs*, not necessarily strictly related to the careers domain, but inherently relevant to Aviator (ultimately, a knowledge graph in its own right). Wikipedia is one of the leaders in this category, given the successful exploitation of Wikidata, an ontology used to extract connections between concepts in various languages. Since the data is not explicitly exposed to the end-user, there is an overall scarcity of programmatic interfaces to the Wikipedia ontologies [46]. The Google Knowledge Graph [41] adds a semantic layer to the classic search engine (the right hand side menu next to the Google search results list is functionally similar to an Aviator sub-graph). However, clicking on a node only displays local information, without expanding the search to another view. On the other hand, Google Knowledge Graph is a powerful, broad spectrum tool, whereas the Aviator ontology is topical in the field of careers, thus better suited to resolve specific queries. A study [44] of how knowledge diversity influences the retrieval of specific ontology data, in the presence of a size restriction, has a possible application for phase two of our platform. Link strengths may be used to define the distance between concepts, thus providing a metric to measure diversity. An excellent survey of techniques for building, mining and expanding knowledge graphs [7] is exemplified on Freebase, the ontology behind Google's Knowledge Graph. Graph Query by Example [23] is a system that uses knowledge samples as a starting point for building queries, in an effort to simplify their structural complexity (illustrated on Freebase and DBpedia). Finally, various relation extraction techniques are suggested [49] in an attempt to transform data from linguistic resources such as WordNet into knowledge graphs.

In summary, the reviewed ontology based career support platforms target narrow professional domains (such as ICT), provide guidance that mainly consists in advanced semantic search features and allow limited support for incorporating non-expert input. In contrast, Aviator employs an ontology spanning over several career fields and offers a rich set of features (career graph navigation and editing, exploration history tracking, etc.). Knowledge graph contributions provide some visualisation of the underpinning ontology, yet are either too specific [7,44,49] or designed for too broad a domain of applications (Google Knowledge Graph) to match the flexibility (node expansion, community edits) and customisation (personal ontologies) of Aviator.

9 Conclusion

Aviator is a career knowledge management platform that is relevant to the more general context of automatic knowledge curation. It exposes the underlying data in navigable, editable views of manageable size, accepts external edits after consistency verification and offers personalised snapshots of users' engagement with the system. This enables Aviator to provide a flexible (as broad or as specific as desired) *perspective* of the field, as opposed to classical career advice platforms where the subtle connections between professions, jobs, educational resources, etc. are obscured by the sheer volume of provided data. Additionally, the Aviator ontology reflects the views of a larger *community* than that of domain experts and also provides the means to *customise* the career researching experience of its end users.

Aviator is powered by a hybrid architecture where semantic tools address knowledge consistency and retrieval issues, whilst the autonomic components manage ontology changes. In this setting, the ontology has several roles: it aligns community curated information (one form of alignment is storing synonyms for each ontology concept), it supports the rendering of relevant knowledge in the form of a navigable graph and it ensures the logic correctness of the knowledge model, via reasoner performed classification.

From an architectural standpoint, Aviator has been analysed with ATAM in order to expose the process behind selecting the most suitable platform design with respect to stakeholders' requirements. The analysis revealed that the structure involving an autonomic manager informed by an ontology met the relevant system goals better than the second best architecture, featuring a NoSQL database to support the knowledge layer.

Future work will be directed towards gathering and analysing Aviator user data (tag usage, graph exploration trends, personal ontology evolution). This will enable fur-

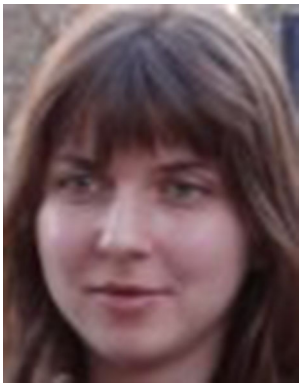
ther platform validation as well as help study part of the career-interested community's dynamics. The latter outcome may prove useful for formulating education/training policies and labour force recruitment strategies. The second planned development is related to the introduction of numerical weights to model the strength of node connections, e.g., "science" is tightly connected to "physics" (link strength 100) but loosely connected to "astrology" (link strength 5). The analyse phase of the MAPE-K control loop will use these weights to display only the nodes connected via strong links to a search keyword. In a broader context, the possibility of applying the autonomic knowledge curation approach embodied by Aviator to other representative domains (e.g., Pinterest) will also be investigated.

Acknowledgements The authors would like to thank Good Careers Guide, Codevate and Capgemini for their continuous support.

References

1. Alimam, M.A., Seghiouer, H., Elyusufi, Y.: Building profiles based on ontology for career recommendation in e-learning context. In: International Conference on Multimedia Computing and Systems (ICMCS), 2014, pp. 558–562. IEEE (2014)
2. Allasia, W., Falchi, F., Gallo, F., Meghini, C.: Autonomic preservation of access copies of digital contents. Digitalization and preservation. In: Proceedings of the Memory of the World in the Digital Age (2012)
3. Allison, P.M.: Dealing with legacy data—an introduction. Internet Archaeol. (2008). doi:10.11141/ia.24.8
4. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Sci. Am. **284**(5), 28–37 (2001)
5. Bigus, J., Schlosnagle, D., Pilgrim, J.R., Mills III, W., Diao, Y.: Able: a toolkit for building multiagent autonomic systems. IBM Syst. J. **41**(3), 350–371 (2002)
6. Blomqvist, E.: The use of semantic web technologies for decision support-a survey. Semant. Web **5**(3), 177–201 (2014)
7. Bordes, A., Gabrilovich, E.: Constructing and mining web-scale knowledge graphs: Kdd 2014 tutorial. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1967–1967. ACM (2014)
8. Carvalho, M.B., Bellotti, F., Berta, R., De Gloria, A., Gazzarata, G., Hu, J., Kickmeier-Rust, M.: A case study on service-oriented architecture for serious games. Entertain. Comput. **6**, 1–10 (2015)
9. Chin, K.L., Chang, E.: A sustainable ict education ontology. In: Digital Ecosystems and Technologies Conference, pp. 350–354 (2011)
10. Cunningham, S.J.: Mining flikr for museum feedback: case study on the qatar islamic museum of art. In: Qatar Foundation Annual Research Conference (2013). doi:10.5339/qfarf.2013.sshp-035
11. Cyganiak, R., Wood, D., Lanthaler, M.: Resource description framework (RDF): concepts and abstract syntax. In: World Wide Web Consortium, January (2013)
12. Fairall, C., Edmunds, H., Cave, D.: BFI national archive: digital workflow for the preservation of digital cinema packages. J. Digit. Media Manag. **2**(2), 127–136 (2013)
13. Faniyi, F., Bahsoon, R., Evans, A., Kazman, R.: Evaluating security properties of architectures in unpredictable environments: a case for cloud. In: 9th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2011, pp. 127–136. IEEE (2011)
14. Ferguson, W.: New geothermal data system could open up clean-energy reserves (2013). https://www.scientificamerican.com/article/new-geothermal-data-system/
15. Fu, B., Noy, N.F., Storey, M.A.: Indented tree or graph? A usability study of ontology visualization techniques in the context of class mapping evaluation. In: International Semantic Web Conference, pp. 117–134. Springer (2013)
16. Gemmell, J., Bell, G., Lueder, R.: Mylifebits: a personal database for everything. Commun. ACM **49**(1), 88–95 (2006)
17. Hall, C., Zarro, M.: Social curation on the website pinterest.com. Proc. Am. Soc. Inf. Sci. Technol. **49**(1), 1–9 (2012)
18. Hazelden, K., Yee-King, M., d'Inverno, M., Confalonieri, R., De Jonge, D., Amgoud, L., Osman, N., Prade, H., Sierra, C., et al.: Wecurate: Designing for synchronised browsing and social negotiation. In: The First International Conference on Agreement Technologies (2012)
19. Huang, C.Y., Chen, R.C., Chen, L.S.: Course-recommendation system based on ontology. In: Machine Learning and Cybernetics, vol. 3, pp. 1168–1173. IEEE (2013)
20. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing—degrees, models, and applications. ACM Comput. Surv. (2008). doi:10.1145/1380584.1380585
21. IBM: an architectural blueprint for autonomic computing. Tech. rep., IBM (2005). http://www-03.ibm.com/autonomic/pdfs/AC
22. Jacob, B., Lanyon-Hogg, R., Nadgir, D.K., Yassin, A.F.: A practical guide to the IBM autonomic computing toolkit. IBM, International Technical Support Organization, Durham (2004)
23. Jayaram, N., Khan, A., Li, C., Yan, X., Elmasri, R.: Querying knowledge graphs by example entity tuples. (2013). arXiv:1311.2100
24. Kasyanov, V., Kasyanova, E.: Information visualisation based on graph models. Enterp. Inf. Syst. **7**(2), 187–197 (2013)
25. Kazman, R., Klein, M., Clements, P.: ATAM: method for architecture evaluation. Tech. rep., DTIC Document (2000)
26. Kephart, J., Chess, D.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
27. Kephart, J.O.: Research challenges of autonomic computing. In: Proceedings of the 27th International Conference on Software Engineering, pp. 15–22. ACM (2005)
28. Leavitt, N.: Will NoSQL databases live up to their promise? Computer **43**(2), 12–14 (2010)
29. Letier, E., Stefan, D., Barr, E.T.: Uncertainty, risk, and information value in software requirements and architecture. In: Proceedings of the 36th International Conference on Software Engineering, pp. 883–894. ACM (2014)
30. Lewis, P.R., Chandra, A., Faniyi, F., Glette, K., Chen, T., Bahsoon, R., Torresen, J., Yao, X.: Architectural aspects of self-aware and self-expressive computing systems: from psychology to engineering. Computer **48**(8), 62–70 (2015)
31. Lopez, V., Fernández, M., Motta, E., Stieler, N.: Poweraqua: Supporting users in querying and exploring the semantic web. Semant. Web **3**(3), 249–265 (2012)
32. Nahar, J., Imam, T., Tickle, K.S., Chen, Y.P.P.: Computational intelligence for heart disease diagnosis: a medical knowledge driven approach. Expert Syst. Appl. **40**(1), 96–104 (2013)
33. Närman, P., Franke, U., König, J., Buschle, M., Ekstedt, M.: Enterprise architecture availability analysis using fault trees and stakeholder interviews. Enterp. Inf. Syst. **8**(1), 1–25 (2014)

34. O'Reilly, T.: What is web 2.0. O'Reilly Media, Inc., Sebastopol (2009)
35. Pandey, P., Misra, R.: Digitization of library materials in academic libraries: issues and challenges. J. Ind. Intell. Inf. (2014). doi:10.12720/jiii.2.2.136-141
36. Patelli, A., Lewis, P., Wang, H., Nabney, I., Bennett, D., Lucas, R., Cole, A.: Autonomic curation of crowdsourced knowledge: the case of career data management. In: 2016 International Conference on Cloud and Autonomic Computing (ICCAC), pp. 40–49. IEEE (2016)
37. Simbulan, M.S.R.: The challenge of managing archaeological databases: some issues and concerns. Hukay **19**, 167–191 (2014)
38. Simperl, E., Luczak-Rösch, M.: Collaborative ontology engineering: a survey. Knowl. Eng. Rev. **29**(01), 101–131 (2014)
39. Singto, P., Mingkhwan, A.: Semantic searching it careers concepts based on ontology. Journal of Advanced Management Science (2013). doi:10.12720/joams.1.1.102-106
40. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: a survey. Semant. Web **3**(2), 169–209 (2012)
41. Steiner, T., Verborgh, R., Troncy, R., Gabarro, J., Van de Walle, R.: Adding realtime coverage to the google knowledge graph. In: 11th International Semantic Web Conference (ISWC 2012) (2012)
42. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The neon methodology for ontology engineering. In: Ontology Engineering in a Networked World, pp. 9–34. Springer (2012)
43. Subotic, I., Rosenthaler, L., Schuldt, H.: A distributed archival network for process-oriented autonomic long-term digital preservation. In: Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries, pp. 29–38. ACM (2013)
44. Sydow, M., Pikua, M., Schenkel, R.: The notion of diversity in graphical entity summarisation on semantic knowledge graphs. J. Intell. Inf. Syst. **41**(2), 109–149 (2013)
45. Taheriyan, M., Knoblock, C., Szekely, P., Ambite, J.L., Chen, Y.: Leveraging linked data to infer semantic relations within structured sources. In: Proceedings of the 6th International Workshop on Consuming Linked Data (COLD 2015) (2015)
46. Torsten Zesch, C.M., Gurevych, I.: Extracting lexical semantic knowledge from wikipedia and wiktionary. In: Nicoletta Calzolari, B.M., Choukri, K. (eds.) Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08) (2008)
47. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: system description. In: International Joint Conference on Automated Reasoning, pp. 292–297. Springer (2006)
48. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The enterprise ontology. Knowl. Eng. Rev. **13**(01), 31–89 (1998)
49. Uszkoreit, H., Xu, F.: From strings to things sar-graphs: a new type of resource for connecting knowledge and language. In: NLP-DBPEDIA@ ISWC (2013)
50. Venters, C., Lau, L., Griffiths, M., Holmes, V., Ward, R., Jay, C., Dibsdale, C., Xu, J.: The blind men and the elephant: towards an empirical evaluation framework for software sustainability. J. Open Res. Softw. (2014). doi:10.5334/jors.ao
51. Weber, N., Schoefegger, K., Bimrose, J., Ley, T., Lindstaedt, S., Brown, A., Barnes, S.A.: Knowledge maturing in the semantic mediawiki: a design study in career guidance. In: Cress, U., Dimitrova, V., Specht, M. (eds.) Learning in the Synergy of Multiple Disciplines. Lecture Notes in Computer Science, vol. 5794, pp. 700–705. Springer, Berlin, Heidelberg (2009)
52. Zwaneveld, B.: Structuring mathematical knowledge and skills by means of knowledge graphs. Int. J. Math. Educ. Sci. Technol. **31**(3), 393–414 (2000)



Alina Patelli is in the final stages of obtaining her most recent Ph.D. in semantic technologies with a special interest in autonomic computing applications. Within the Aston Lab for Intelligent Systems Engineering (ALICE), she is also doing research relevant to evolutionary programming and self-aware systems.

Peter R. Lewis is a Lecturer in Computer Science at Aston University in the UK, and is a member of the Aston Lab for Intelligent Collectives Engineering (ALICE). He has a background in bio-inspired computing, and his research is concerned with continuous adaptation and learning in complex agent-based systems.

Aniko Ekart holds a Ph.D. in Computer Science from Eötvös Loránd University, Budapest, Hungary. She is a Senior Lecturer at Aston University. She has been Head of Computer Science between August 2016 and May 2017 and postgraduate programme director in Computer Science since 2008. She is a member of the Aston Lab for Intelligent Collectives Engineering (ALICE). Her research interests are in theory and application of evolutionary computation, artificial intelligence and data mining. She has worked on a series of European projects applying these methods in engineering, transport logistics and heath domains.

Hai Wang obtained his B.Sc. in 2001 and Ph.D. in 2003 from the National University of Singapore. He has authored over 50 papers and articles with over 100 non-self citations on the areas of Semantic Web, Knowledge Engineering and Software Engineering. He has served as a program committee member for over 20 international conferences. His research on applying Web and Semantic Web technologies to formal knowledge sharing won the IEEE IT Book

Prize. After working within the EPSRC-funded project HyOntUse (Hybrid User Oriented to Ontology Tools) and JISC-funded project CO-ODE (Collaborative Open Ontology Development Environment), he and his colleagues from the University of Manchester and Stanford University successfully developed the Protégé-OWL system, which has become the de facto standard environment for building OWL ontologies internationally. In the EU-IST Project TAO (Transitioning Applications to Ontologies), he was responsible for leading Southampton's contribution. His focus was on developing the methodology and the tools to make transitioning existing 'legacy' knowledge and applications to ontologies quickly and effectively. He became a Lecturer at Aston University in 2009.

Ian Nabney is 50th Anniversary Chair of Systems Analytics. His research spans both the theory and applications of neural networks and other pattern recognition techniques, with a special focus on data visualisation and probabilistic modelling. Much of his work is inspired, directly or indirectly, by industrial problems in bioinformatics, biosignal processing, condition monitoring, remote sensing and financial forecasting. He has put his experience of software engineering to good use through developing the Netlab toolbox for neural networks and related pattern analysis techniques: this has now been downloaded more than 45,000 times and the accompanying book has 1000 citations. The current focus of his research is in data visualisation (representing high-dimensional data faithfully in 2D so that users can analyse its structure visually) and time series analysis and characterisation (with applications in biomedical signal processing and condition monitoring of complex machinery).

David Bennett graduated from Aston University and is co-founder of Codevate, a successful middleware consultancy company in the Birmingham area.

Ralph Lucas read Physics at Balliol College, Oxford and since then has worked as a chartered accountant, investment banker, publisher and politician. Has maintained an active interest in data and programming throughout.

Alex Cole specialises in working to redefine public services and develop socially innovative solutions to enhance and support local citizens (particularly those excluded from mainstream engagement socially, financially, and digitally) to play a more active role in their communities development through improved wellbeing, skills, and lifestyle learning. He has a particular interest in the piloting and research of the innovative uses of Blockchain within smart city frameworks and standards, as a mechanism to identify and capture the intrinsic and extrinsic value of learning to achieve traceable, autonomous, and provenance based social impact which can support economic regeneration of high deprivation communities and places.