Journal of Internet Services and Applications

RESEARCH Open Access

An open source library to parse and analyze online collaborative knowledge-building portals

Amit Arjun Verma*, S.R.S Iyengar, Simran Setia and Neeru Dubey

*Correspondence:
2016csz0003@iitrpr.ac.in
[1]Social Computing and Collective Intelligence Lab, Indian Institute of Technology Ropar, Rupnagar, India

Abstract

With the success of collaborative knowledge-building portals, such as Wikipedia, Stack Overflow, Quora, and GitHub, a class of researchers is driven towards understanding the dynamics of knowledge building on these portals. Even though collaborative knowledge building portals are known to be better than expert-driven knowledge repositories, limited research has been performed to understand the knowledge building dynamics in the former. This is mainly due to two reasons; first, unavailability of the standard data representation format, second, lack of proper tools and libraries to analyze the knowledge building dynamics.

We describe Knowledge Data Analysis and Processing Platform (KDAP), a programming toolkit that is easy to use and provides high-level operations for analysis of knowledge data. We propose Knowledge Markup Language (Knol-ML), a generic representation format for the data of collaborative knowledge building portals. KDAP can process the massive data of crowdsourced portals like Wikipedia and Stack Overflow efficiently. As a part of this toolkit, a data-dump of various collaborative knowledge building portals is published in Knol-ML format. The combination of Knol-ML and the proposed open-source library will help the knowledge building community to perform benchmark analysis.

Link of the repository: Verma et al. (2020)

Video Tutorial: Verma et al. (2020)

Supplementary Material: Verma et al. (2020)

Keywords: Knowledge building, Wikipedia, Stack exchange, Open-source, Python library

1 Introduction

With progress in computational power, research in various domains is primarily based on the availability of data and appropriate tools for analysis. Open access to libraries and data enhances the ease and pace of research [1]. The impact of open-source tools (like Python, R, and Scilab) can be verified by the expansion in the utility of these tools by the research community [2]. For example, a simple task like matrix inversion requires multiple lines of code to be written in Python. Whereas, the usage of NumPy library reduces the complexity of this task to a single line of code. Similar examples can be found in various domains, where the usage of analysis tools reduces the complexity of tasks in terms of time and effort. It is useful to note that in recent years, the scientific community is positively influenced by a growing number of libraries, such as scikit-learn for machine learning, NumPy and SciPy for statistical computing, and matplotlib for visualization [3].

The advancement in computational power and storage facilities allows crowdsourced portals, such as Wikipedia, Stack Overflow, Quora, Reddit, and GitHub, to host their data on publicly available servers. The popularity and open access to the datasets of these crowdsourced portals have drawn the attention of researchers from various communities. Observing the collaboration and contribution of the crowd, researchers have generalized the knowledge development on these portals to the actual knowledge building process [4]. From predicting box office success of movies to building state-of-the-art software, these portals have helped the research communities in various aspects [5–7].

The diverse and rich content present on Wikipedia is used to study online collaboration dynamics [8, 9], to examine its impact on other online collaborative portals [10], and to train state-of-the-art artificial intelligence algorithms [11]. Similarly, the massive growth of users and posts on crowdsourced QnA portals like Stack Overflow, Yahoo! Answers[1], and Quora, have attracted the attention of researchers to study the dynamics of knowledge building on these portals [12–16]. Adding to the portal-specific analyses of knowledge building on wiki-based portals and QnA portals, inter-portal analyses would stimulate an ecosystem to give a broader perspective of knowledge building dynamics. Recent literature has shown a rise in demand for understanding the relationships between these portals [10, 17].

The unique functionalities of wiki-based portals and QnA portals have resulted in different representation formats of their datasets. For research involving the large-scale analyses of these portals, there is an underlying problem of the unavailability of datasets in a standard format at one place. Furthermore, the existing tools for data extraction and analysis are narrowly focused on specific portals. For example, finding the contributors who are common to Wikipedia and Stack Overflow requires multiple steps of pre-processing and separate analysis of the corresponding datasets.

Goal: Based on the fact that a large fraction of researchers in the scientific community are dependent on software tools [18, 19], we aim to create an ecosystem by standardizing the representation of the datasets of collaborative knowledge-building portals and providing a toolkit to analyze these datasets.

Although there is an abundance of portal-specific APIs (e.g. Wikipedia API [20], Wikidata Query Service [21], Stack Exchange API [22], Reddit API [23]), the bottleneck is the restriction on API calls. Another downside is the requirement to learn different APIs for performing analyses on different portals. The structural difference of the extracted data from crowdsourced portals necessitates intermittent pre-processing for analyses. We emphasize the absence of a standard representation format for the data of these portals, albeit the commonality of extensive knowledge development. While there has been substantial research using the dataset of these portals, none of the existing tools have the potential to fulfill the requirements mentioned above. Our work will act as a catalyst for knowledge building research by bridging the gap between the supply of data and demand for analysis.

[1] Although we mention Yahoo! Answers as a QnA portal, it will be set to read-only mode on April 20th, 2021, and no longer be usable in its current form

This manuscript extends our earlier work [24]. The significant changes are as follows: 1) We evaluate our toolkit based on scalability and generalizability on a new dataset of JOCWiki; 2) We modify the Wiki-based compression method to store in the revision-history dataset in Knol-ML format; 3) We discuss the limitations of our toolkit in terms of Knol-ML representation and generalizability of KDAP methods; 4) We provide more details on the crowdsourced portals in the Background Section 2.

The key contributions of this paper are as follows:

1 An XML [25]-based Knowledge Markup Language (Knol-ML), a novel approach for standardizing the representation of dataset of various collaborative knowledge building portals.
2 Knowledge Data Analysis and Processing Platform (KDAP), a standard high-performance library that is easy to use and provides high-level operations for the analysis of knowledge data.
3 Knowledge Analysis and Retrieval (KAR) dataset, a set of data of these portals in Knol-ML format.

KDAP is developed for single big-memory multiple-core machines and stabilizes the disparity between maximum performance and compact in-memory data representation. KDAP aims at facilitating the knowledge building community with open access to standard dataset and efficient data analysis methods. This will increase the ease of inter-portal analyses and also reduce the overhead for researchers to handle different data formats and corresponding APIs/libraries for analyses of these portals.

The rest of the paper is organized as follows. First we discuss the functionalities of Wiki-based portals and QnA-based portals (Section 2). Following the details, we present the relevant literature, including the various representation formats and tools to analyze the dataset of knowledge building portals (Section 3). Then we describe the Knol-ML format and the extension mechanism associated with it (Section 4). The details of the KDAP library is provided in the next section (Section 5). Finally, we present the evaluation of KDAP and results of the case studies performed (Section 6).

2 Background

Prior to related work, we provide a high-level context about online collaborative portals and how knowledge matures on these portals. Majority of the online knolwledge portals are collaborative in nature. However, they do not represent all the knowledge building portals. For instance a blog article is written and maintained by a single user, making the article non-collaborative in nature. We first define the online crowdsourced portals based on the degree of freedom provided to users on each knowledge instance. We then classify these portals into two categories. The first class of portals is where people contribute towards the development of comprehensive information on a topic, based on resources present on the Internet. Wikipedia and Wikia are instances of such portals. The second category is a set of portals, where structured discussion is the core feature, opening the ground for asking questions and answering them. Unlike Wikipedia, knowledge in these portals is built according to the point of view of the users. In the upcoming subsections we define the online crowdsourced portals and each of its categories.

2.1 Defining "Crowdsourced" portals

We define the online crowdsourced portals based on the foundation laid by Anamika et al. [26]. The authors defined crowdsourced portals as platforms where users have the freedom of either modifying a knowledge unit or adding a new knowledge unit on top of the previous one. Moreover, every user is allowed to add the knowledge units in the portal[2]. This degree of freedom allows the connection among the knowledge units, creating a well-connected knowledge base. Interestingly, not all online portals provide this degree of freedom to their users. For instance on Amazon Mechanical Turk, only a specific set of users are allowed to perform a task (based on the client's requirement). Based on the above definition, we could classify the crowdsource portals into two categories; Wiki-based (Wikipedia, GitHub, Wikia) and QnA-based (Stack Exchange, Quora, Reddit). In the rest of the paper, we use crowdsourced portals and collaborative knowledge building portals interchangeably.

2.2 Wikipedia: a portal with revisions

Wikipedia is built on the idea that "Anyone Can Edit", aided with functions like *history* and *edit*. The rigorous usage of these functions help an article reach a mature state, as *history* allows the users to trace the development and *edit* allows them to add their perspective to the article. Articles in Wikipedia follow *non-point of view* style, demanding users to support their contribution with relevant evidence in the form of references. Crowd collaboration, rich knowledge, and open data dump has made Wikipedia a popular hub for conducting research in various domains. The success of Wikipedia has sowed the seeds of Wiki technology owing to the prosperity of wiki-based portals (e.g. Wikia).

2.3 QnA portal: portal for structured discussion

Another category of crowd-sourced portals is Question and Answer portals (e.g. Stack Overflow, Quora), where the crux is the generation of exhaustive structured discussion on a particular topic. People perform various tasks including asking questions, providing answers, discussing a topic, upvoting or downvoting a post, etc. These functionalities facilitate the crowd to generate question specific knowledge. The increasing usage of QnA portals and the availability of data has led researchers to explore knowledge building dynamics on these portals.

3 Related work

In this section, we present the relevant literature regarding standard data representation formats in various domains as well as open-source tools to analyze the data.

3.1 Data representation and analysis

The problem of standard data representation and tools to analyze data exists in other related domains. In particular, the dynamic nature of graph structures claims for a standard representation format for visualization and analyses. This claim has been addressed with the formulation of formats, such as Pajek, Graph Modeling Language (GML), Graph Markup Language (GraphML), and Graph Exchange XML Format (GEXF) [27]. Further, these formats have triggered the development of open-source tools like NetworkX,

[2]In some portals (like Stack Exchange), users are required to gain some experience before able to add/modify every knowledge unit.

iGraph, SNAP, and Gephi for analysis and visualization of graphs [28, 29]. The availability of such standard formats and analysis tools promotes open science and helps in the advancement of scientific research [30].

3.2 Tools and parsers for crowdsourced portals

Though portal-specific tools are developed to present the analyses of questions asked for particular crowdsourced portals, standard tools, regardless of the portal, are unavailable. We now discuss a few of these standard tools in the following subsections.

3.2.1 Wikipedia-based APIs

Various tools and libraries have been developed to analyze Wikipedia data. Most of these tools extract the data in real-time to answer questions. A common example of such a tool is the web-based Wikipedia API. It has been frequently used to retrieve language links [31], finding articles related to a word [32], getting details about edits in articles [33], etc. However, the downside of using a web-based API is that a particular revision has to be requested from the service, transferred over the Internet, and then stored locally in an appropriate format. Setting up a local Wikipedia mirror helps circumvent the drawback, but the constraint that Wikipedia API provides full-text revisions leads to a large amount of data being transferred. A similar tool is the Wikipedia Query Service [21], which helps user query against the wikidata [34] dataset. Currently, in the beta mode of its development, it contains a list of complex queries (e.g. Metro stations in Paris, places that are below 10 meters above sea level), which can be asked on the live dataset. Though the tool is sophisticated, there is a stipulated deadline of 60 seconds, beyond which queries cannot be asked. Such tools are useful for sampling data from the entire set or asking specific queries, but the flexibility with data and algorithms is limited.

3.2.2 Wikipedia data parsers

Apart from web-based services, tools to extract and parse the Wikipedia data dump are available. Ferschke et al. [35] developed the Wikipedia Revision Toolkit, which represents Wikipedia revisions in compressed form by just storing the edits, hence decreasing the size of the dump by 98%. Efficient retrieval methods were implemented to retrieve the data of a particular revision from the compressed data dump. Wikibrain, developed by Sen et al. [36], is another example of Wikipedia data dump parser, which surmounts the unavailability of standard Wikipedia-based algorithms. It provides state-of-the-art algorithms ranging from extracting pageviews to finding semantic relatedness, but the number of algorithms is limited. The usage of various parsers for wiki markup-based [37] data dumps (e.g. MediaWiki Utilities [38], mwxml [39]) are restricted to basic data.

3.2.3 QnA-based tools

Few QnA-based portals provide APIs for users to query the database. For instance, Stack Exchange API [22] developed by Stack Exchange network is employed to obtain various analyses results. Similarly, API provided by Reddit (Reddit API [23]) can be used to acquire information like user details, subedits, etc. Like Wikipedia APIs, these APIs also have a limit on the number of calls restricting large-scale analyses.

4 Knol-ML: knowledge markup language

Analysis of crowdsourced knowledge building portals entails data storage to facilitate easy retrieval and exchange. Such portals vary in their functionalities, resulting in a distinct schema for storing data in the database. Due to the structural difference of the schemata, data dump provided by these portals have different representation formats. The steps involved in the process of analyzing the data are; data retrieval, pre-processing, and analysis. The absence of standard libraries and tools to analyze the data of crowdsourced portals follows as yet another downside, restricting the knowledge building community to analyze and study the dynamics of these portals.

Motivated by the goal of having a standard library and access to benchmark datasets for crowdsourced portals, we propose Knowledge Markup Language (Knol-ML), an XML based data representation format for these portals. We specify the following:

1. Core elements to represent the knowledge data structures.
2. An extension mechanism that allows to independently add portal-specific knowledge data with the base as core elements.

The extension mechanism provides a feature of adding extra information that can be combined or ignored without affecting the overall structure. Thus, a portal specific format can be created, respecting the core structure of Knol-ML. The heterogeneity in the representation formats of the data of different portals has created a dire need for the feature mentioned above. There are various data serialization formats like JSON, YAML, SOAP, and XML, of which we found XML [40] the best fit for developing standard data representation format for crowdsourced portals. The reason being its descriptive tagging structure, where each knowledge unit can be defined separately and the support for the extension.

4.1 Core layer

Sequential knowledge development (Fig. 1) is the fundamental feature of crowdsourced knowledge building portals, where the crowd develops knowledge sequentially by adding, deleting, and editing information [26]. Over and top of this fundamental feature, some portals have customized features like upvoting and downvoting others' contributions, accepting answers, and adding comments. In this section, we describe the representation of the data of crowdsourced portals in Knol-ML format. The core layer constitutes the foundation of the format, describing the underlying elements to represent sequential

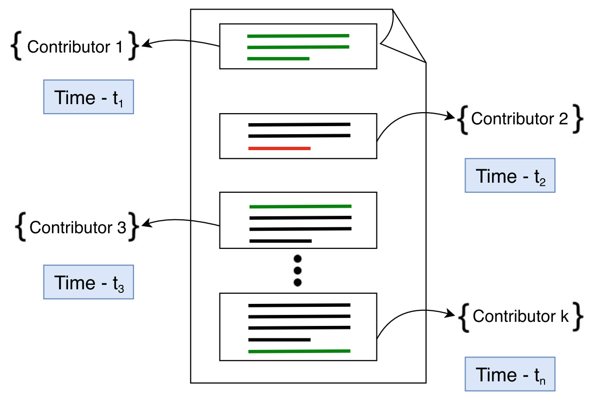


Fig. 1 An illustration of *sequential knowledge* building in crowdsourced environment. A contribution can be seen in the form of addition (green) or deletion (red) information

knowledge development. It can also include additional information (e.g., user's biography), which can be easily identified and optionally ignored, leaving the structural information of the core layer undisturbed. This extension is by virtue of extensibility provided by the XML framework. Section 4.2 Extension Mechanism defines additional information in this context, and its inclusion.

We define *instance* as a piece of information which is added by a *contributor* at a particular *timestamp*. The idea is to break the knowledge data into a series of *instances*, making it the fundamental unit of our format. This fragmentation allows us to represent the dataset of any knowledge building portal as a set of instances, each having a set of attributes like contributor, timestamp, and text. The idea is motivated by the GraphML format, where a set of vertices and edges defines a graph, with each having its own set of parameters.

We define our format based on the activities performed on the collaborative knowledge building portals. As explained earlier, users exhibit various activities in the knowledge development process, as illustrated in Fig. 1. Each document is considered as separate *Knowledge Data*, which may contain multiple instances of edits performed by various contributors. The structure depicted in Fig. 1 constitutes the fundamental structure of our format. The XML schema of the Knol-ML format has been provided in the Resources section.

Figure 2 depicts the Knol-ML format for sequential knowledge data. A Knol-ML document may contain multiple `<KnowledgeData>`, each representing different document. The topic of a document is described using the `<Title>` tag. The information regarding a particular edit performed by a contributor is described within the `<Instance>` tag. It contains three mandatory tags explained below:

- A `<TimeStamp>` tag contains a sub-tag `<CreationDate>`, which stores the commit date and time of this edit.
- `<Contributors>` tag stores the details regarding editor of this instance. It has two further sub-tags, `<OwnerUserName>`, which stores the name of the editor and `<OwnerUserId>`, which stores the unique Id assigned to the editor. The former is optional, but the latter is mandatory.
- The `<Body>` tag contains a sub-tag `<Text>`, which stores the actual edits of this instance.

```
<KnowledgeData Type="text" Id="1">
    <Title>This is an example for sequential knowledge data</Title>

    <Instance Id="1" InstanceType="Revision">
        <TimeStamp><CreationDate>t1</CreationDate></TimeStamp>
        <Contributors>
            <OwnerUserName>Editor 1</OwnerUserName>
            <OwnerUserId>1</OwnerUserId>
        </Contributors>
        <Body><Text Type="text">Edits of Editor 1</Text></Body>
    </Instance>

    <Instance Id="2" InstanceType="Revision">
        <TimeStamp><CreationDate>t2</CreationDate></TimeStamp>
        <Contributors>
            <OwnerUserName>Editor 2</OwnerUserName>
            <OwnerUserId>2</OwnerUserId>
        </Contributors>
        <Body><Text Type="text">Edits of Editor 2</Text></Body>
    </Instance>
          ⋮
</KnowledgeData>
```

Fig. 2 Representation of sequential knowledge data in Knol-ML format

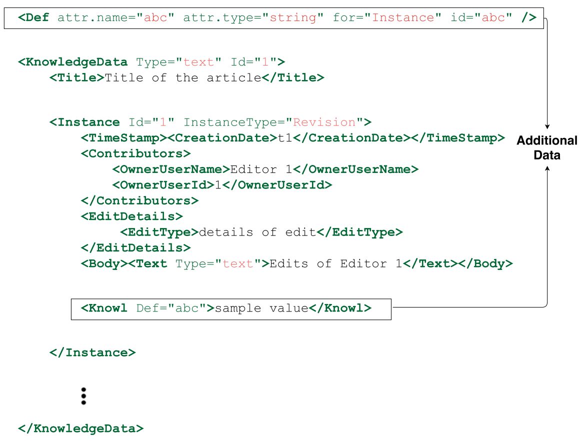
The sub-tags of <Contributors> and <Body> tags ensure that additional information can be included within them appropriately.

The knowledge building portals can be divided into two categories, *wiki*-based and *question and answer (QnA)*-based. Although some of the knowledge building portals (e.g., Github) cannot be classified into either of these two categories, their data can be represented in Knol-ML format following the sequential knowledge model with the help of an extension mechanism. For wiki-based portals, each article is considered as <KnowledgeData> in Knol-ML format whereas, for QnA-based portals, a *thread* is considered as <KnowledgeData> which is defined as a question and all the posts (answers, comments, votes, scores, etc.) related to it [41]. The full description of the data representation of wiki-based and QnA-based portals is present in the supplimentry material [57] (Appendix Wikipedia revision history compression and Getting started with KDAP).

4.2 Extension mechanism

The core layer defined previously describes the generalized structure of sequential knowledge in Knol-ML format. We use the extension mechanism, a feature of XML, to represent the additional portal specific information (e.g., badges' details, users' bio, pageviews, and editors' location). Brandes et al. [42] have used a similar approach in GraphML for representing additional information in graphs. They have defined the location of the additional data with the help of < key> and < data> tags. New data can be accommodated within the < data> tag, which requires a < key> providing a name and domain (type and range of values it can take) of this new data. The domain is defined using `for` attribute of the < key> tag.

Analogously, we have defined <Knowl> and <Def> tag to represent the additional information, as shown in Fig. 3. The <Def> tag can be defined by including attributes name, domain, and location of this new information. The name, domain, and location are defined using `attr.name`, `attr.type` and `for` attribute respectively. Each element in the Knol-ML core layer can contain multiple <Knowl> tags, representing the additional information for that element. Using this extension mechanism, any additional information can be described in the Knol-ML format. Also, a parser need not support



```
<Def attr.name="abc" attr.type="string" for="Instance" id="abc" />

<KnowledgeData Type="text" Id="1">
    <Title>Title of the article</Title>

    <Instance Id="1" InstanceType="Revision">
        <TimeStamp><CreationDate>t1</CreationDate></TimeStamp>
        <Contributors>
            <OwnerUserName>Editor 1</OwnerUserName>
            <OwnerUserId>1</OwnerUserId>
        </Contributors>
        <EditDetails>
             <EditType>details of edit</EditType>
        </EditDetails>
        <Body><Text Type="text">Edits of Editor 1</Text></Body>

        <Knowl Def="abc">sample value</Knowl>

    </Instance>

        ⋮

</KnowledgeData>
```

Fig. 3 Representation of extension mechanism in Knol-ML

the extensions to extract information from the core structure. All the `<Def>` tags can be safely ignored as they appear before the `<KnowledgeData>` tag, which is the root of every sequential knowledge data.

5 KDAP implementation details

KDAP is a system designed for analyzing knowledge data represented in Knol-ML format. KDAP design allows the methods of analysis to work on any category of knowledge data. The foundational classes of KDAP are centered around Knol-ML format. These classes are categorized into *wiki-based containers* and *QnA-based containers*. Wiki-based containers, `WRKnol` and `WRCKnol`, represent the revision based wiki data. `WRKnol` corresponds to the full revision and `WRCKnol` corresponds to the compressed form. In the case of QnA-based portals, the data may or may not be available in the form of revisions. Hence, `QKnol` corresponds to the knowledge data, which is not in the form of revisions. `QRKnol` corresponds to knowledge data with revisions.

The idea is to optimize the execution time and memory usage of these methods by choosing the appropriate container class. The containers provide different methods for knowledge data, including revision and non-revision based. The standardized structure of Knol-ML simplifies the implementation of a new algorithm, as each algorithm has to be implemented once, which can be executed on any knowledge data. This implementation helps in comparing different genres of crowdsourced portals under the umbrella of knowledge building.

Methods implemented on wiki and QnA-based containers can be divided into three categories; knowledge *generation* methods, for the generation of new knowledge data (which is stored in Knol-ML format); knowledge *conversion* methods, for the conversion of data into Knol-ML format and knowledge *analytic* methods, for the computation of specific knowledge building-related analysis without manipulating the underlying structure. As future work, we will also include manipulation methods that can be used to modify knowledge data.

5.1 Containers functionality

KDAP provides a standard interface for both the containers, bypassing the step of preprocessing the data of multiple knowledge building portals individually. Every method in KDAP is built on top of KDAP *iterators* that provides a container-independent traversal of the instances of knowledge data. The KDAP converters can be used to club all the `<KnowledgeData>` into a single file, or multiple files can be created, each having the same number of `<KnowledgeData>`. The design of creating multiple files allows parallel processing without compromising the dependency. Thus, enabling a massive data dumps to be broken down into multiple chunks which can be loaded efficiently, optimizing the memory usage. Also, the excessive increase in the number of files can be avoided by including multiple `<KnowledgeData>` in a single Knol-ML file.

5.2 Knowledge data representation

KDAP has been designed based on the Knol-ML structure. Hence, it is essential to have a data structure such that accessing and analyzing methods are computationally efficient. For example, accessing a particular instance of a Wikipedia article or Stack Overflow thread should be reasonably fast and not expensive in terms of memory consumption.

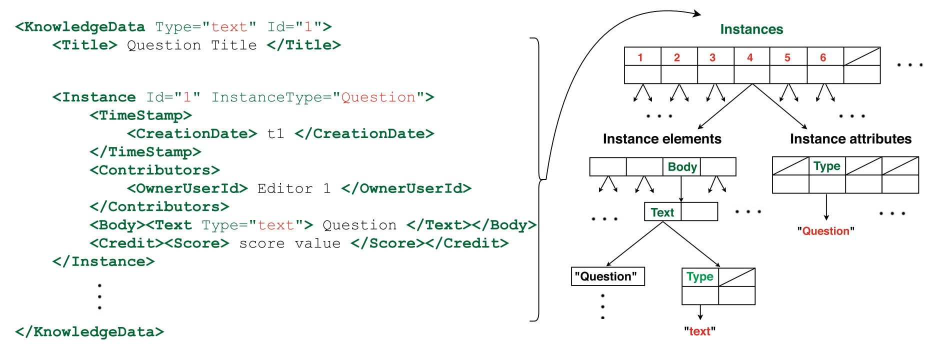
Handling these features is imperative, as the data of these portals will increase many folds with time. The trade-off between time and space calls for a representation that optimizes both of these. Furthermore, in collaborative knowledge building portals order of knowledge arrival should be preserved. In general, ordering is achieved by using vector-based representation, while speed is achieved by using hash table-based representation.

For KDAP, we have chosen a middle ground between all-hash-table and all-vector-knowledge representations. A hash table has been used to represent the instances of knowledge data. The idea behind using the hash table is to reduce the time complexity for retrieving an instance, which plays a crucial role in processing the knowledge data. Each instance consists of a unique identifier, a single hash table storing the attributes of the instance, and a vector representing the elements of the instance. Elements of the vector may further contain a hash table and a vector representing its attributes and elements, respectively. This cross between the hash table and the vector is designed to store the hierarchical data structure of Knol-ML in the memory. Figure 4 summarizes knowledge representations in KDAP.

5.3 Time complexity of key operations

For the analysis of knowledge data, atomic operations must be efficient and less time-consuming. KDAP allows the algorithms to work on small chunks of data at a time, reducing the overall memory consumption and time complexity. This is achieved by the help of the Knol-ML structure, which allows the KDAP parser to process one instance at a time. Furthermore, it gives an additional advantage of processing the knowledge data parallelly, providing an extra benefit of time complexity reduction. Since most of the operations are dependent on the retrieval of information from Knol-ML data like contributors' information, and time details, we have focused on optimizing the time and space complexity of such operations.

To process the Knol-ML files efficiently, its optimal representation in RAM is a prerequesite. Wikipedia, being a revision based system, stores all the edits performed by the editors, resulting in the accumulation of a lot of redundant information. Owing to this, the size of each article reaches megabytes or even gigabytes. To reduce the overall space complexity of processing revision based Knol-ML files while optimizing the time complexity, we compress the revision-based Knol-ML by efficiently storing the differences between



```
<KnowledgeData Type="text" Id="1">
    <Title> Question Title </Title>


    <Instance Id="1" InstanceType="Question">
        <TimeStamp>
            <CreationDate> t1 </CreationDate>
        </TimeStamp>
        <Contributors>
            <OwnerUserId> Editor 1 </OwnerUserId>
        </Contributors>
        <Body><Text Type="text"> Question </Text></Body>
        <Credit><Score> score value </Score></Credit>
    </Instance>
            ⋮

</KnowledgeData>
```

Fig. 4 A diagram of knowledge data structures. Instance ids are stored in a hash table, and each instance has one hash table and one vector associated with it representing the elements of the instance

Table 1 Time Complexity of Key Operations in KDAP

Operations	Time Complexity[a]	
	Wiki-Based	QnA-Based
Get an instance	$\mathcal{O}(k^2m)$	$\mathcal{O}(n)$
Get all instances	$\mathcal{O}(kmn)$	$\mathcal{O}(n)$
Retrieve instance attributes	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Add an instance	$\mathcal{O}(1)$	$\mathcal{O}(1)$

[a]The time complexity of $\mathcal{O}(1)$ is achieved by hashing the instances. Although hashing has worst case time complexity of $\mathcal{O}(n)$, on an average $\mathcal{O}(1)$ complexity is achieved.

consecutive revisions. A naive algorithm will require to store the edits made in the current revision exclusively[3], but the revision access time will increase as each revision will now require reconstructing the text from a list of changes. Thus, to accelerate the reconstruction process, every k^{th} revision is stored as a full revision, where $k << n$ (n is the total number of revisions). A similar method is proposed by Ferschke et al. [35], where the value of each interval (k) is fixed to 1000 irrespective of the article size. We modify Ferschke's algorithm by keeping the interval length k as a function of n for each article. Our experiment on a sample of Wikipedia full revision history dataset revealed the optimal value of k as $\sqrt{n}$, optimising both time and space complexity. The details of the experiment is provided in the Appendix (Wikipedia revision history compression). However, in KDAP, the user has an option of tuning the value of k.

Table 1 shows the time complexity of key operations on knowledge data in KDAP. Here, k is the number of instances that are stored between two successive full revisions, and n is the total number of instances present in the knowledge data. As described before, the size of $k << n$, which means that the time complexity of an instance retrieval in case of wiki-based data is very less. This is because the size of the compressed wiki-based data is considerably small as compared to the actual XML dump, which allows KDAP methods to store it in RAM. Similarly, the size of a thread in a QnA portal is very less as compared to the total number of posts in the portal, providing an extra benefit of storing multiple threads parallelly in memory. As we show in the evaluation section, KDAP can provide high performance in terms of memory and time complexity while allowing for efficient execution of algorithms.

Figure 5 is an overview of the KDAP framework, which summarizes the entire process of Knol-ML conversion and analysis.

6 Evaluation

In KDAP, we have implemented commonly used traditional algorithms as well as new methods for analyzing and understanding the dynamics of collaborative knowledge building. All these methods accept Knol-ML files as data input and return the analysis results. The main disadvantage with collaborative online portals is that the data dump is in raw format and requires multiple steps of cleaning for analysis purposes. With the design of Knol-ML and KDAP, we have created a system that reduces the time and effort required to retrieve and analyze knowledge data. We provide many atomic level methods which can be used as building blocks for analysis such as language statistics as a measure of quality of contribution [44–47], global inequality as a measure of participation [48, 49],

[3]Using the *difference* of the current revision with the previous one using the diff algorithm [43]

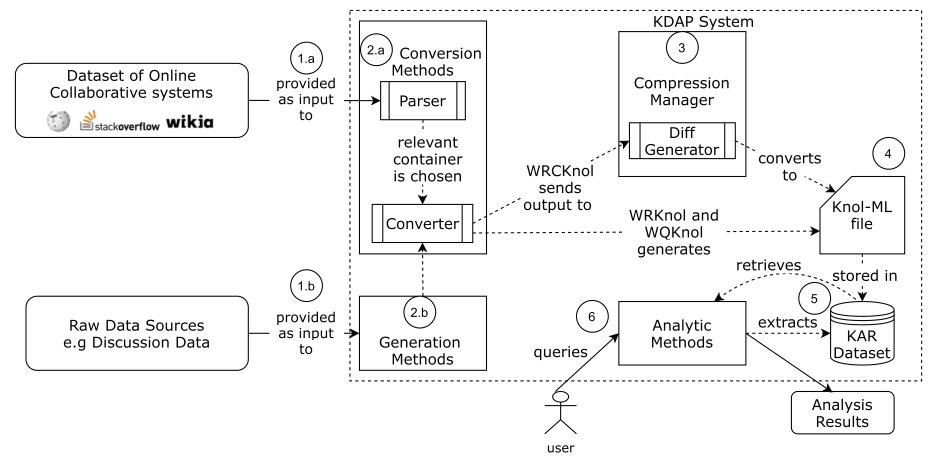


Fig. 5 Overview of KDAP framework. First, the relevant dataset is extracted from the source based on the requirement (1.a and 1.b). Depending on the type of the dataset, relevent containers are used to convert the dataset in Knol-Ml format (2.a and 2.b). The compression manager is called to efficiently store the dataset in Knol-ML format (3). The Knol-ML file is generated which is stored on the KAR dataset repository (4). The KAR dataset acts as the main repository for extracting and retrieving the relavent information (5). A user only uses the KDAP analysis methods to extract, parse, and analyze the relevant dataset stored in the KAR repository (6). We regularly maintain the KAR repository, containing the current version of the datasets

editing patterns to understand collaboration [50–52], data extraction for various machine learning and NLP algorithms [11, 53, 54].

To evaluate our tool, we describe two evaluation methodologies with the following goals in mind:

- To show that our toolkit performs better than the present analysis toolkits based on the parameters such as execution time, memory consumed, the complexity of the code, and the lines of code.
- to show that it is possible to perform complex analyses using our toolkit without significant effort.

The first evaluation methodology includes six common mining tasks, whereas the second methodology includes large-scale analysis tasks. The tasks listed in both the methodology were performed twice, including and excluding KDAP. The authors in [55] have used a similar approach to evaluate the PyDriller tool with other existing tools. The tasks in the first evaluation methodology were designed to measure our tool based on the *comparision with present tools*, whereas the tasks of the second methodology evaluate the *usefulness* of our tool for large-scale analyses. We describe and compare the analysis of both the evaluation methodologies. There is no such library for the large-scale analysis of collaborative knowledge-building portals to the best of our knowledge. Hence, we compare the performance of KDAP with existing libraries and APIs commonly used for extracting and parsing the dataset of these portals. All the analyses were performed on a computer with a 3.10GHz Intel Xeon E5-1607 processor and 16 GB of RAM. The analyses were performed five times, and the average execution time and memory consumption are shown.

6.1 Evaluation based on comparision with present tools

We first compare KDAP against existing libraries and APIs for online collaborative portals. We select six commonly known knowledge building tasks that we encountered in

Table 2 Tasks assigned to the first group

Data Extraction	
Task 1	Extracting 5 Wikipedia articles from each quality category namely FA, GA, B, C, Start and Stub.[a]
Task 2	Extracting 10,000 random questions, its answers and comments from Stack Exchange site, say, anime.stackexchange.com
Data Parsing	
Task 3	Finding the number of words, sentences and Wikilinks added/deleted in each revision of an article (United States).
Task 4	Extracting all the questions which had an accepted answer from anime.stackexchange.com
Analysis Methods	
Task 5	Find the correlation between monthly pageviews and the number of revisions of an article (United States).
Task 6	Find the correlation between Gini coefficient (a measure of inequality of contribution) and answer to question ratio for various stack stackexchange portals.

[a]Wikipedia has defined seven quality grades, starting from Stub class to Featured Articles (FA) class, where the least developed articles are in class Stub and fully developed articles are in FA class

our experience as researchers in the knowledge building field. We divided the tasks into three groups, as shown in Table 2. The reason behind this segregation is to evaluate our tool based on a variety of tasks commonly performed by the knowledge building researchers. We compare the analyses using different metrics: lines of code (LOC), complexity (McCabe complexity [56]), memory consumption, and execution time of both implementations. Table 3 shows the results. We do not count the number of lines for the code which do not contribute to the core functionality (like constructor). Instead, we use a fixed lines of code value of 3 for all such codes.

Regarding execution time, KDAP is 63.32 and 8.33 times faster than the respective tool for tasks 1 and 2, respectively. This speed is achieved because KDAP maintains a database of Wikipedia articles name and corresponding categories (please see Appendix for more details). For other tasks, the performance of KDAP is similar to that of other tools. In terms of memory consumption, the tools behave similarly. In most of the cases, memory consumption was less than 20MB. In the most memory consuming task (task 4), 86MB of memory was used. Given the massive size of the dataset (e.g. the size of United States articles is close to 6 GB) and limited main memory size, we had to performed iterative

Table 3 Comparison between KDAP and various libraries

		Time (sec)		**Memory (MB)**		**Complexity**	**LOC**
		mean	**std**	**mean**	**std**		
Task-1	KDAP	**7.13**	0.51	**0.98**	0.34	**2**	**7**
	A	70.45	1.21	4.07	0.97	8	33
Task-2	KDAP	**2.91**	0.81	**15**	1.3	**6**	**19**
	T	11.24	1.17	16.2	1.01	13	88
Task-3	KDAP	**521**	12.32	2.01	0.46	**1**	**5**
	T & P	528	13.63	**1.9**	0.21	2	120
Task-4	KDAP	4.8	0.47	86	1.60	**2**	**9**
	T	**2.7**	0.31	**83.2**	2.14	4	36
Task-5	KDAP	**80**	1.56	**0.42**	0.13	**1**	**8**
	A	81.2	1.70	0.69	0.27	4	45
Task-6	KDAP	86	2.39	**2**	0.71	**2**	**13**
	T	**82.3**	2.59	4.3	0.63	4	70

A, T and P refers to Wikipedia API, cElementTree and mwparserfromhell respectively. The boldface values represent the parameters on which KDAP outperforms the other listed libraries parsing (for tasks 3, 4, 5, and 6) of the files while using Wikipedia API, cElementTree and mwparserfromhell. More precisely, we iterate over each block of a file and process it, keeping the overall memory consumption constant. This iterative parsing adds to the complexity of the code. KDAP by default provides method to iteratively parse the files, where a user has the freedom to either load a whole file in the memory or process it chunkwise.

The more significant difference is in terms of the complexity of the implementation and LOC. For the former, we observe that using KDAP results (on average) in writing 61% less complex code as compared to respective libraries. This is specially the case in tasks that have to deal with mining Wikipedia and Stack Exchange (Task1 and 2); indeed, obtaining this information in KDAP is just a one line code, while Wikipedia API and Stack Exchange API require many lines of code and exceptions handling.

We also observe that the number of lines of code written using KDAP is significantly lower than for the respective library. Table 3 shows that, on an average, 84% fewer lines of code are required using KDAP. The biggest difference is in task 3, where the tool had to calculate the change in words, sentences and Wikilinks for each revision of an article. This problem was solved in five LOC using KDAP, while 120 LOC with cElementTree (95% difference). The details of the experiment are provided in the supplimentry material [57] (Appendix Getting started with KDAP), and codes for all the implementations are available in the experiment section of the GitHub repository.

6.2 Evaluation based on usefulness

To further analyze our tool, we choose four peer-reviewed articles in the CSCW domain to be analyzed using KDAP. We took the help of four undergraduate students working in the knowledge building domain to re-perform the analysis mentioned in these articles. Each participant was assigned one paper, as shown in Table 4. They were asked to perform the analyses twice (including and excluding KDAP) and note the time they took to solve the problems, as well as their personal opinions on all the tools. All the students had an experience in developing with Python and on performing knowledge building studies, but they had never used KDAP before. The setting of the experiment is the following:

- Each participant is assigned one paper, which he/she has to implement first with KDAP, then with any other library of their choice. Since understanding how to solve the tasks requires some additional time, we asked the participants to start with KDAP. This choice clearly penalizes our tool, as participants will have a better intuition about the tasks during the first round of implementation. However, we believe that KDAP is simpler to use and that the difference between the two implementations will still be significant.
- For the sake of simplicity, participants should only implement the core analysis methods. Methods like machine learning model training and graph plotting were excluded.

Table 4 Papers Assigned to the Participants

Participant	Paper Assigned
P1	Black Lives Matter in Wikipedia: Collaboration and Collective Memory around OSM [58]
P2	Crowd Diversity and Performance in Wikipedia [59]
P3	An Empirical Study on Developer Interactions in StackOverflow [60]
P4	Improving Low Quality Stack Overflow Post Detection [61]

- Participants note the time taken to implement the tasks. They are also asked to include the time spent in reading the documentation of the tools, since understanding how to use the tool is part of the experiment.
- After having implemented all the tasks, we ask the participants to elaborate on the advantages and disadvantages of the tools.

The result of the experiment is shown in Table 5. All the participants took less time to solve the problems (26% less in the worst case, 71% less in the best case). Regarding the LOC, three out of four participants wrote significantly less LOC. P4, instead solved both problems using a similar amount of time and LOC: the participant first solved the problem using KDAP and applied the same logic to solve the problem using `cElementTree`.

All the participants agreed that KDAP was more comfortable to use than other analysis tools (P1 to P4). For example, P1 affirmed that using KDAP, he was able to achieve the same result with more straightforward and shorter code, and that he will continue to use KDAP in his subsequent knowledge building studies. P2 added that Wikipedia and Stack Exchange APIs are useful when one has to perform limited extraction tasks, but it can be overcomplicated when the goal is to perform broad-scale analysis on these portals, for which KDAP is more appropriate because it hides this complexity from the users.

6.3 Evaluation based on scalability and generalizability: a case study of JOCWiki

Although we have established our toolkit's usefulness, it is essential to show if our toolkit is scalable enough to analyze a new portal's dataset. Moreover, it is interesting to evaluate the generalizability of the analysis methods using cross-portal analysis.

To understand the extent of scalability and generalizability of our toolkit, we describe the analysis of JOCWiki with KDAP as a case study. JOCWiki is an online crowdsourced portal that follows a unique integration of a Wiki-like portal and a discussion-styled forum [62]. JOCWiki was deployed as a part of a MOOC named "The Joy of Computing" to create a rich knowledge repository using the course students as the contributors. The authors showed that this integration of Wiki and QnA forum (also known as QWiki) generates more knowledge as opposed to a single disintegrated platform. The integration of Wiki and discussion-forum like features in a single platform allows us to evaluate our

Table 5 Time and Loc Comparison for Each Participant

Participant			With KDAP	Without KDAP	Total
P1	Time (minutes)	mean	70	190	-63%
		std	2.12	5.7	
	LOC		19	401	-95%
P2	Time (minutes)	mean	110	390	-71%
		std	14.13	19.17	
	LOC		50	292	-82%
P3	Time (minutes)	mean	46	90	-48%
		std	3.72	3.42	
	LOC		34	83	-60%
P4	Time (minutes)	mean	52	60	-13%
		std	5.30	7.28	
	LOC		81	91	-10%

Table 6 Evaluation of KDAP methods based on scalibility and generalizability. Intra-portal analysis tasks were performed on both Wiki and QnA forum of the JOCWiki, whereas inter-portals analysis tasks represent the cross portal analysis between Wiki and QnA forum

Task	**KDAP**		**Without KDAP**	
	LOC	**Complexity**	**LOC**	**Complexity**
Intra-Portal Analysis				
Number of Editors	3	1	23	3
Number of edits/posts	3	1	23	3
Edits/posts by Year/Month/Day	3	1	28	4
Words in each edit/post	5	1	120	2
External links in each edit/post	5	1	120	2
Number of posts and its comments	🟥 14	🟥 2	86	2
Sentiment of each edit/post	8	1	127	2
Extracting Author's age/country/bio	🟥 27	🟥 3	76	3
Measure of contribution in Wiki/QnA	3	1	143	3
Inter-Portal Analysis				
Intersection of edits and post by month/year	19	2	119	3
Intersection of editors in Wiki and QnA	5	1	146	3
Wiki-articles refereed in QnA forums and vice versa	57	5	186	8
Topics from Wiki refereed in QnA forums and vice versa	74	5	203	8

🟥represents the functions are not generalizable.

toolkit based on scalability and generalizability. We show that our toolkit can handle the dataset of JOCWiki, and the fundamental analysis can be performed in lesser lines of code. We also show the generalizability of our toolkit by performing same tasks on Wiki and QnA forum of JOCWiki.

We extracted the full dataset of JOCWiki, publically available at GitHub [63]. The dataset contained twelve weeks of Wiki articles (one for each course module) and their respective discussion forum in a raw XML and text format, respectively. We first converted the dataset of Wiki articles and their respective discussion forum into Knol-ML format. Although the conversion step is an overhead in terms of time and computation, it is essential since it enables a user to execute all the KDAP library methods on the converted Knol-ML dataset. We extended the KDAP library by adding the JOCWiki conversion methods to it. We perform a set of fundamental analysis on the JOCWiki Knol-ML dataset using KDAP and without KDAP[4]. We divide the tasks into two categories. The first category of tasks requires similar analysis on both the portals (Wiki articles and their respective QnA forums), and hence we perform them on both the portals (Wikis and QnAs) separately. The second category contains a set of tasks that require cross-portal analysis between the Wiki articles and the QnA forums. We aim to determine if we can perform the mentioned analysis with less complex codes while using KDAP. We compare the results based on LOC (Lines of Code) and Complexity (McCabe complexity [56]). Table 6 represents the result of the experiment.

6.3.1 Results on scalibility

We observed that using KDAP, we could perform the analysis in lesser lines of code compared to using the conventional libraries. More precisely, we could perform all the mentioned tasks using average 95% lesser lines of code with KDAP compared to using conventional libraries. Moreover, with our toolkit, we could write less complex codes.

[4]We used conventional libraries as mentioned in subsection 6.1

We observed that the fundamental methods of the KDAP library are useful in performing cross-portal analysis using less complex and fewer lines of code. For instance, extracting topics from Wiki articles that triggered discussion in QnA forums and vice versa is comparatively more straightforward with KDAP. This quantification of triggers was a significant contribution by Simran et al. [62] where they showed that the QWiki setup generates more knowledge units than conventional collaborative knowledge building systems. The reason behind writing more complex codes while using the conventional libraries was the unstrctured formatting of QnA forum dataset. Retireving information from an unstructured dataset is generally difficult and time consuming. Given the standardized structure of the Knol-ML format, such information retrieval and analysis tasks are easy to perform with KDAP.

6.3.2 Results on generalizability

In terms of generalizability, we observed that most of the KDAP methods were generalizable over both the portals while performing intra-portal analysis tasks. More specifically, for most of the tasks, common KDAP methods were used to analyze both Wiki articles and QnA forums. We had to use a different set of methods for some tasks as no common methods were applicable (red colored rows in Table 6 represent those tasks). For example, to extract the author's information (such as age, country, and bio) from both Wiki articles and QnA forums, we had to use the external file of author's dataset. One solution for this problem is to include all author's detailed information in the Knol-ML dataset. However, including such information will increase the size of a single Knol-ML `KnowledgeData` (refer to Fig. 2 for details) by many folds. We discuss this tradeoff in detail in the Limitations and future work section.

6.4 Comparison of KDAP with other tools

There are various tools like WikiBrain, DBpedia, and Wikipedia API to analyze the knowledge data. Although these tools provide analysis and retrieval methods, knowledge building analysis methods (like edit statistics, inequality measure, and controversy detection) are limited in number. Also, these tools are limited to specific portals. KDAP provides exclusive methods for analyzing the dynamics of collaborative knowledge building. We define a set of tasks (defined in Table 7) based on which we compare our tool with other analysis tools. Table 8 shows a comparison of methods implemented in KDAP with the other analysis tools.

7 Resources

KDAP library is publically available at GitHub [64]. The git repository contains user documentation, tutorials (with an introductory video) [65], Knol-ML schema, stable releases of KDAP, and supplementary material [57]. As a part of KDAP, we are also maintaining the Knowledge Analysis and Retrieval (KAR) dataset, containing the knowledge dataset of different collaborative portals, including Wikipedia (all Featured articles and Good articles), Stack Exchange network, and Wikia in Knol-ML format. More datasets are being collected and will be updated in the future. KDAP source code has been released under a permissive BSD type open-source license. Being an open-source library, we welcome the community to contribute to KDAP and KAR dataset. Please refer to the Appendix for details regarding Getting started with KDAP.

Table 7 Tasks defined to compare KDAP with other analysis tools

Tasks	Alias
Wikipedia article extraction by name	Task-1
Wikipedia article extraction by class	Task-2
Stack Exchange extraction by portal name	Task-3
Pageviews extraction of Wikipedia/Stack Exchange	Task-4
Link/Category/Image extraction	Task-5
Revision/text extraction	Task-6
Contributors' details extraction	Task-7
Question/Answers/Comments extraction	Task-8
Semantic Relatedness	Task-9
Edit Statistics	Task-10
Measure of Inequality	Task-11
Controversy detection	Task-12

8 Limitations and future work

Although we have focused on creating a standard representation format and analysis toolkit, there are a few limitations in this approach. First, converting an unstructured dataset into the Knol-ML format may increase the final Knol-ML dataset size. This increase in the size is mainly due to the extra tags required to represent the dataset in Knol-ML format. For instance, converting the JOCWiki QnA dataset into the Knol-ML format increases the QnA dataset's size by 17%. However, Knol-ML representation allows the KDAP methods to extract the data efficiently.

Secondly, while most of the fundamental methods are generalizable over different datasets, having portal specific methods is inevitable. This specificity of methods sometimes requires the writing of more complex codes to perform portal specific tasks. Moreover, the generalizability of KDAP methods comes from the Knol-ML representation format, which sometimes leads to the accumulation of extra information (for

Table 8 Comparison of KDAP methods with other analysis tools

Methods	KDAP	WikiBrain	JWPL[a]	DBpedia[b]	Wikipedia API	SE API
Extraction Methods						
Task-1	🟩 Yes	🟩 Yes	🟩 Yes	🟥 No	🟧 Yes	🟥 No
Task-2	🟩 Yes	🟥 No	🟥 No	🟥 No	🟧 Yes	🟥 No
Task-3	🟩 Yes	🟥 No	🟥 No	🟥 No	🟥 No	🟧 Yes
Task-4	🟩 Yes	🟧 Yes	🟥 No	🟥 No	🟧 Yes	🟧 Yes
Parsing Methods						
Task-5	🟩 Yes	🟩 Yes	🟩 Yes	🟩 Yes	🟧 Yes	🟧 Yes
Task-6	🟩 Yes	🟩 Yes	🟩 Yes	🟩 Yes	🟧 Yes	🟥 No
Task-7	🟩 Yes	🟥 No	🟥 No	🟥 No	🟧 Yes	🟧 Yes
Task-8	🟩 Yes	🟥 No	🟥 No	🟥 No	🟥 No	🟧 Yes
Knowledge Building Methods						
Task-9	🟥 No	🟩 Yes	🟥 No	🟩 Yes	🟥 No	🟥 No
Task-10	🟩 Yes	🟥 No	🟥 No	🟥 No	🟥 No	🟥 No
Task-11	🟩 Yes	🟥 No	🟥 No	🟥 No	🟥 No	🟥 No
Task-12	🟩 Yes	🟥 No	🟥 No	🟥 No	🟥 No	🟥 No

🟧 Yes represents that the number of calls are limited.
[a] https://dkpro.github.io/dkpro-jwpl/
[b] https://wiki.dbpedia.org/

example, adding new tags to identify the information uniquely), eventually increasing the overall size of the dataset. Finally, to use the KDAP methods on a new portal's dataset, the data must first be converted into Knol-ML format, an overhead in time and computation. However, the conversion task is a one time process as we are maintaining a public repository (KAR dataset) containing all the dataset in Knol-ML format. Users can directly download a portal's dataset in Knol-ML format from the KAR repository.

As future work, we aim to develop more functions for the KDAP library. The library is kept open for the researchers/developers to use and extend. Also, based on the generalized structure, visualization methods can be implemented to visualize the knowledge data. Moreover, some of the KDAP methods can be implemented in the C/C++ programming language, optimizing the overall running time and memory usage. We encourage readers to install KDAP and explore its functionalities; most importantly, we encourage researchers and developers to improve KDAP and KAR dataset by joining our team on GitHub.

9 Conclusion

We have developed Knol-ML, an XML-based standard representation format for the data of crowdsourced knowledge building portals. We also provide KDAP, a standard high-performance system, to analyze the data of these portals. With Knol-ML and KDAP methods, complex analysis can be performed using a few lines of code. Apart from Knol-ML and KDAP, we release the KAR dataset, which contains the data of crowdsourced knowledge building portals in Knol-ML format. KDAP provides basic as well as advanced retrieval and analysis methods that have been proven useful in studying collaborative knowledge building portals. We believe that the combination of Knol-ML, KDAP, and KAR Dataset as a toolkit will accelerate the research in knowledge building community.

Appendix

Wikipedia revision history compression

We extend the fixed-interval length ($k = 1000$) method provided by Ferschke et al. [35] and develop an online algorithm that calculates the interval length (k) based on the input file, optimizing both the time and space complexity. We experiment on a sample of Wikipedia articles to find the optimal value of k. We experiment on different values of the interval lengths ranging from $k = 2$ to $k = n - 1$, including the interval length proposed by Ferschke et al. [35]. The reason behind choosing these interval lengths is to experimentally find the optimal compression, minimizing both extraction time and space complexity. We perform the comparison based on revision's random access time and compressed to original document ratio. Since a single revision's random access time is minimal, we calculate the access time of 100 random revisions and take the aggregate. For each article in the dataset, we calculate all the two parameters and present the mean and standard deviation.

Dataset

The data collection was conducted by using retrieval methods of KDAP (Knowledge Data Analysis and Processing Platform) toolkit. A stratified sampling was conducted on four Wikipedia quality classes: Featured Articles (FA), Good Articles (GA), Class B articles, and Class C articles. We excluded Stub and Start class articles from our sampling as these articles comparatively have fewer revisions - meaning there is no potential requirement

Table 9 Number of articles sampled from each class

Class	Number of Articles
Featured Articles	32
Good Articles	162
Class B	487
Class C	1202

for compression. A-class contains a few set of articles which we merged with GA-class while performing the sampling. Since the majority of the Wikipedia articles with a number of revisions greater than 1000 fall in one of the mentioned categories, these four classes were chosen to cover the articles with more considerable lengths. Similar sampling approaches have been taken to cover articles from various topics [59, 66]. For each article in the data set, its complete editing history in Knol-ML format was collected between the article's creation time to November 2019. The initial sampling resulted in 2000 articles. Among them, 118 articles had very few revisions (less than 100). These were excluded from the sampling, leaving the final data set of 1882 articles.

Each Knol-ML document contains an article's full edit history with additional information such as contributor's id, comments, and time stamp. Each instance (revision) in a Knol-ML document has an id tag, which always starts from one for the first instance. We leverage this information to perform random access on the compressed dataset. Table 9 presents the number of articles sampled from each class.

Results

Table 10 shows the aggregate random retrieval time and compression ratio for different k values. The algorithm using $k = \sqrt{n}$ outperforms the fixed k method (Ferschke et al. [35]) by a large margin. We achieve an optimal random revision extraction time without compromising much on the compression ratio for all k values. More precisely, we achieve an average random retrieval time of 0.080 seconds, which is 16 times faster than the fixed k method ($k = 1000$). Moreover, we achieve the average compression ratio of 0.174, a ratio only 1.64 times more than the ratio observed using the fixed k method. We achieve a comparatively low standard deviation in terms of random revisions extraction time, indicating our method's stable performance over all the articles. However, in terms of compression ratio, we achieve a larger standard deviation. A large standard deviation is mainly because for smaller articles (revisions less than 500), our method estimates a comparatively smaller interval length, increasing the compression ratio.

Getting started with KDAP

Table 10 Comparison of compression methods using various interval lengths based on revision retrieval time and compression ratio

Interval Length	Time (seconds) Random Revision Extraction		Memory Original to Compressed Ratio	
	avg	std	avg	std
$k = 2$	0.008	0.011	0.566	0.075
$k = \sqrt{n}$	0.090	0.271	0.166	0.113
$k = 1000$	1.290	1.65	0.114	0.096
$k = n - 1$	2.254	5.163	0.103	0.091

In this section we provide a brief demonstration about how to use KDAP.

Installation

KDAP is deployed on pip and can be installed using the command `pip install kdap`. The library is present on GitHub and can be installed from there. The GitHub repository will be updated after the acceptance. Please refer to the supplementary material [57] (Section 2) for more information.

Downloading dataset

Using kdap is very simple. You only need to create the `knol` object which can be further used to call the kdap methods. For example, the following lines of code will download and store the full revision history of Wikipedia article *Gravity* in the desired directory.

```
import kdap
knol = kdap.knol()
knol.get_wiki_article('Gravity',[output_dir])
```

Similarly, user can download the stack exchange portals data using the following lines of code:

```
import kdap
knol = kdap.knol()
stack_list = ['3dprinting', 'ai', 'arduino', 'boardgames', 'chemistry', 'chess']
for portal in stack_list:
    knol.download_dataset(sitename='stackexchange', portal=portal)
```

Extraction

Sampeling dataset from Wikipedia or Stack Exchage requires only a few lines of code. For example, suppose you want random five articles from each category of Wikipedia classes:

```
import kdap
knol = kdap.knol()
from random import sample

category_list = ['FA', 'GA', 'B', 'C', 'Start', 'Stub']
articles = {}
for category in category_list:
    articles[category] = sample(knol.get_wiki_article_by_class
    (wiki_class=category), 5)
```

Frame-Wise analysis

After downloading the relevant Knol-ML dataset, various analysis methods can be used. To perform more granular level analysis, one can retrieve the instances of a Knol-ML file as `frames` and use the frame-based analysis methods. For instance, the following lines of code extract the Wikipedia article's revision as frames and find information such as, contributor, time-stamp, score, etc.:

```
import kdap
knol = kdap.knol()

knol.get_wiki_article('Gravity')
frame = knol.frame(file_name='Gravity.knolml')

for each in frame:
    print(each.get_editor()) #prints each revision's editor's name and unique id
    print(each.get_timestamp()) #prints the timestamp of each revision's creation
     date
    print(each.get_score()) #prints the score (upvotes, downvotes) associated with
      each revision, if present
    each.get_text(clean=True) #returns the clean text of each instance
```

Complex analysis

KDAP provides high-level complex methods to perform detailed analysis on the knowledge building dataset. For example, a comparison of knowledge building portals based on Global Gini coefficient which require multiple steps of processing can be easily performed using KDAP by writing the following lines of code:

```
import kdap
knol = kdap.knol()
stack_list = [‘english’, ‘superuser’, ‘askubuntu’, ‘gaming’, ‘diy’, ‘tex’]
gini_list = []
atoq_ratio = []
for stackportal in stack_list:
    knol.download_dataset(sitename=‘stackexchange’, portal=stackportal)
 gini_list.append(knol.get_global_gini_coefficient(dir_path=‘
     directory_path_of_data’))
    questions = knol.get_num_instances(dir_path=stackportal+‘/Posts’,
     instance_type=‘question’)
    answers = knol.get_num_instances(dir_path=stackportal+‘/Posts’, instance_type
     =‘answer’)
    atoq_ratio.append(questions[‘questions’]/answers[‘answers’])
```

Abbreviations
KDAP: Knowledge Data Analysis and Processing Platform; Knol-ML: Knowledge Markup Language; QnA: Questions and Answers; JOCWiki: Joy of Computing Wiki Portal; LOC: Lines of Code

Acknowledgements
Not applicable.

Authors’ contributions
A.A. Verma collaborated in the problem formalization, created the KDAP library, performed the experiments to evaluate the library, participated in forming the KAR dataset, and participated in the manuscript writing. S.R.S. Iyengar coined the original problem definition, supervised the library development, and supervised the manuscript writing. S. Setia collaborated to analyze results obtained from the experimental evaluations and collaborated in the manuscript writing. N. Dubey participated in testing the library’s functionalities, collaborated in creating the KAR dataset, and collaborated in the manuscript writing. All authors read and approved the final manuscript.

Funding
A part of this work was funded by the assistance received by Dr. S.R.S Iyengar from CSRI, Department of Science and Technology India via grant no. SR/CSRI/344/2016.

Availability of data and materials
Please refer to the Resources section.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 22 January 2021 Accepted: 16 September 2021
Published online: 02 November 2021

References
1. McKiernan EC, Bourne PE, Brown CT, Buck S, Kenall A, Lin J, McDougall D, Nosek BA, Ram K, Soderberg CK, et al. Point of view: How open science helps researchers succeed. Elife. 2016;5:16800.
2. Harry Thornburg. Four main languages for Analytics, Data Mining, Data Science. 2014. https://www.kdnuggets.com/2014/08/four-main-languagesanalytics-data-mining-data-science.html. Accessed 14 Mar 2019.
3. Zhao R, Wei M. Impact evaluation of open source software: An altmetrics perspective. Scientometrics. 2017;110(2): 1017–33.
4. Cress U, Kimmerle J. A systemic and cognitive view on collaborative knowledge building with wikis. Int J Comput-Supported Collab Learn. 2008;3(2):105.
5. Wang G, Gill K, Mohanlal M, Zheng H, Zhao BY. Wisdom in the social crowd: an analysis of quora. In: Proceedings of the 22nd international conference on World Wide Web. Rio de Janeiro; 2013. p. 1341–52.
6. Mestyán M, Yasseri T, Kertész J. Early prediction of movie box office success based on wikipedia activity big data. PloS one. 2013;8(8):71226.
7. Dabbish L, Stuart C, Tsay J, Herbsleb J. Social coding in github: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. Seattle: ACM; 2012. p. 1277–86.
8. Kittur A, Kraut RE. Beyond wikipedia: coordination and conflict in online production groups. In: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work. Savannah: ACM; 2010. p. 215–24.

9. Chhabra A, Iyengar SS. Characterizing the triggering phenomenon in wikipedia. In: Proceedings of the 14th International Symposium on Open Collaboration. Paris: ACM; 2018. p. 1–7.
10. Vincent N, Johnson I, Hecht B. Examining wikipedia with a broader lens: Quantifying the value of wikipedia's relationships with other large-scale online communities. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. Montreal: ACM; 2018. p. 566.
11. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. Lake Tahoe Nevada: Curran Associates Inc.; 2013. p. 3111–9.
12. Kayes I, Kourtellis N, Quercia D, Iamnitchi A, Bonchi F. Cultures in community question answering. In: Proceedings of the 26th ACM Conference on Hypertext & Social Media. Florence: ACM; 2015. p. 175–84.
13. Dong Z, Shi C, Sen S, Terveen L, Riedl J. War vs inspirational in forrest gump: Cultural effects in tagging communities. In: 6th International AAAI Conference on Weblogs and Social Media, ICWSM 2012. Paris: AAAI; 2012. p. 82–9.
14. Oliveira N, Muller M, Andrade N, Reinecke K. The exchange in stackexchange: Divergences between stack overflow and its culturally diverse participants. Proc ACM Hum Comput Interact. 2018;2(CSCW):130.
15. Quattrone G, Mashhadi A, Capra L. Mind the map: the impact of culture and economic affluence on crowd-mapping behaviours. In: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. Baltimore: ACM; 2014. p. 934–44.
16. Chhabra A, Iyengar S. Activity-selection behavior of users in stackexchange websites. In: Companion Proceedings of the Web Conference 2020. Taipei: ACM; 2020. p. 105–6.
17. McMahon C, Johnson I, Hecht B. The Substantial Interdependence of Wikipedia and Google: A Case Study on the Relationship Between Peer Production Communities and Information Technologies. In: Proceedings of the International AAAI Conference on Web and Social Media, Vol. 11, No. 1. Montreal: AAAI; 2017.
18. Howison J, Bullard J. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. J Assoc Inf Sci Technol. 2016;67(9):2137–55.
19. Nangia U, Katz DS, et al. Track 1 paper: surveying the us national postdoctoral association regarding software use and training in research. In: Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE 5.1); 2017.
20. Mediawiki. Wikipedia API. 2021. https://en.wikipedia.org/w/api.php. Accessed 10 May 2021.
21. Mediawiki. Wikidata Query Service. 2021. https://query.wikidata.org/. Accessed 10 May 2021.
22. Exchange S. Stack Exchange API. 2021. https://api.stackexchange.com/. Accessed 10 May 2021.
23. Reddit. Reddit API. 2021. https://www.reddit.com/dev/api/. Accessed 10 May 2021.
24. Verma AA, Iyengar S, Setia S, Dubey N. Kdap: An open source toolkit to accelerate knowledge building research. In: Proceedings of the 16th International Symposium on Open Collaboration. Madrid: ACM; 2020. p. 1–11.
25. Consortium TWWW. XML. 2021. https://www.w3.org/TR/REC-xml/. Accessed 10 May 2021.
26. Chhabra A, Iyengar S. How does knowledge come by? arXiv preprint arXiv:1705.06946. 2017.
27. Cherven K. Mastering Gephi Network Visualization. Birmingham: Packt Publishing Ltd; 2015.
28. Akhtar N. Social network analysis tools. In: 2014 Fourth International Conference on Communication Systems and Network Technologies. Bangalore: IEEE; 2014. p. 388–92.
29. Leskovec J, Sosič R. Snap: A general-purpose network analysis and graph-mining library. ACM Trans Intell Syst Technol (TIST). 2016;8(1):1.
30. Nosek BA, Alter G, Banks GC, Borsboom D, Bowman SD, Breckler SJ, Buck S, Chambers CD, Chin G, Christensen G, et al. Promoting an open research culture. Science. 2015;348(6242):1422–5.
31. Steiner T, Van Hooland S, Summers E. Mj no more: Using concurrent wikipedia edit spikes with social network plausibility checks for breaking news detection. In: Proceedings of the 22nd International Conference on World Wide Web. Rio de Janeiro: ACM; 2013. p. 791–4.
32. Wu Z, Giles CL. Sense-Aaware Semantic Analysis: A Multi-Prototype Word Representation Model Using Wikipedia. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, No. 1. Austin: AAAI; 2015.
33. Boukhelifa N, Chevalier F, Fekete J-D. Real-time aggregation of Wikipedia data for visual analytics. In: 2010 IEEE Symposium on Visual Analytics Science and Technology. Salt Lake City: IEEE; 2010. p. 147–54.
34. Mediawiki. Wikidata. 2021. https://www.wikidata.org/wiki/Wikidata:Main_Page. Accessed 10 May 2021.
35. Ferschke O, Zesch T, Gurevych I. Wikipedia revision toolkit: efficiently accessing wikipedia's edit history. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations. Portland: Association for Computational Linguistics; 2011. p. 97–102.
36. Sen S, Li TJ-J, Team W, Hecht B. Wikibrain: democratizing computation on wikipedia. In: Proceedings of The International Symposium on Open Collaboration. Berlin: ACM; 2014. p. 27.
37. Mediawiki. Wiki Markup Language. 2021. https://en.wikipedia.org/wiki/Help:Wikitext. Accessed 10 May 2021.
38. Aaron Halfaker. MediaWiki Utilities. 2019. https://pythonhosted.org/mediawiki-utilities/. Accessed 28 Feb 2019.
39. Aaron Halfaker. MediaWiki XML Processing. 2019. https://pythonhosted.org/mwxml/. Accessed 21 Mar 2019.
40. Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F. Extensible markup language (XML). World Wide Web J. 1997;2(4):27–66.
41. Wang GA, Wang HJ, Li J, Abrahams AS, Fan W. An analytical framework for understanding knowledge-sharing processes in online q&a communities. ACM Trans Manag Inf Syst (TMIS). 2015;5(4):18.
42. Brandes U, Eiglsperger M, Herman I, Himsolt M, Marshall MS. Graphml progress report structural layer proposal. In: International Symposium on Graph Drawing. Berlin Heidelberg: Springer; 2001. p. 501–12.
43. Hunt JW. An Algorithm for Differential File Comparison. Technical report, Computing Science Technical Report, Bell Laboratories. 1976.
44. Blumenstock JE. Size matters: word count as a measure of quality on wikipedia. In: Proceedings of the 17th International Conference on World Wide Web. Beijing: ACM; 2008. p. 1095–6.
45. Kumar S, West R, Leskovec J. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In: Proceedings of the 25th International Conference on World Wide Web. Montréal: International World Wide Web Conferences Steering Committee; 2016. p. 591–602.8.
46. Novielli N, Calefato F, Lanubile F. Towards discovering the role of emotions in stack overflow. In: Proceedings of the 6th International Workshop on Social Software Engineering. Hong Kong: ACM; 2014. p. 33–6.

47. Correa D, Sureka A. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In: Proceedings of the 23rd International Conference on World Wide Web. Seoul: ACM; 2014. p. 631–42.
48. Vasilescu B, Filkov V, Serebrenik A. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In: 2013 International Conference on Social Computing. Alexandria: IEEE; 2013. p. 188–95.
49. Arazy O, Nov O. Determinants of wikipedia quality: the roles of global and local contribution inequality. In: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work. Savannah: ACM; 2010. p. 233–6.
50. Liu J, Ram S. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. ACM Trans Manag Inf Syst (TMIS). 2011;2(2):11.
51. Wierzbicki A, Turek P, Nielek R. Learning about team collaboration from wikipedia edit history. In: Proceedings of the 6th International Symposium on Wikis and Open Collaboration. Gdańsk: ACM; 2010. p. 27.
52. Yang J, Hauff C, Bozzon A, Houben G-J. Asking the right question in collaborative q&a systems. In: Proceedings of the 25th ACM Conference on Hypertext and Social Media. Santiago: ACM; 2014. p. 179–89.
53. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013.
54. Treude C, Robillard MP. Augmenting api documentation with insights from stack overflow. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). Austin: IEEE; 2016. p. 392–403.8.
55. Spadini D, Aniche M, Bacchelli A. Pydriller: Python framework for mining software repositories. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Lake Buena Vista: ACM; 2018. p. 908–11.8.
56. McCabe TJ. A complexity measure. 308–20. 1976;4:.
57. Verma AA, Iyengar SRS, Setia S, Dubey N. Supplimentry Material. 2020. https://github.com/kdap/kdap.github.io/blob/master/Appendix.pdf. Accessed 16 Jan 2021.
58. Twyman M, Keegan BC, Shaw A. Black lives matter in wikipedia: Collective memory and collaboration around online social movements. In: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing. Portland: ACM; 2017. p. 1400–12.
59. Ren R, Yan B. Crowd diversity and performance in wikipedia: The mediating effects of task conflict and communication. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Denver: ACM; 2017. p. 6342–51.8.
60. Wang S, Lo D, Jiang L. An empirical study on developer interactions in stackoverflow. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. Coimbra: ACM; 2013. p. 1019–24.
61. Ponzanelli L, Mocci A, Bacchelli A, Lanza M, Fullerton D. Improving low quality stack overflow post detection. In: 2014 IEEE International Conference on Software Maintenance and Evolution. Victoria: IEEE; 2014. p. 541–4.
62. Setia S, Iyengar S, Verma AA. Qwiki: Need for qna & wiki to co-exist. In: Proceedings of the 16th International Symposium on Open Collaboration; 2020. p. 1–12.
63. Setia Simran, Iyengar S. R. S, Verma AmitArjun. The Joy of Computing Dataset. 2020. https://github.com/SimranIITRpr/QWiki-Dataset-. Accessed 16 Jan 2021.
64. Verma AmitArjun, Iyengar S. R. S, Setia Simran, Dubey Neeru. Knowledge Data Analysis and Processing Platform. 2020. https://github.com/descentis/kdap. Accessed 16 Jan 2021.
65. Verma AmitArjun, Iyengar S. R. S, Setia Simran, Dubey Neeru. KDAP Tutorial. 2020. https://www.youtube.com/watch?v=O82hLMsJiUM&list=PLwHIH9xdhJLC2eDvOUzOnyOfxpwOZLCB-. Accessed 16 Jan 2021.
66. Arazy O, Nov O, Patterson R, Yeo L. Information quality in wikipedia: The effects of group composition and task conflict. J Manag Inf Syst. 2011;27(4):71–98.

Publisher's Note