KDAP: An Open Source Toolkit to Accelerate Knowledge Building Research

Amit Arjun Verma
SCCI Labs, IIT Ropar
Rupnagar, India
2016csz0003@iitrpr.ac.in

S.R.S. Iyengar
SCCI Labs, IIT Ropar
Rupnagar, Punjab, India
sudarshan@iitrpr.ac.in

Simran Setia
SCCI Labs, IIT Ropar
Rupnagar, Punjab, India
2017csz0001@iitrpr.ac.in

Neeru Dubey
SCCI Labs, IIT Ropar
Rupnagar, Punjab, India
neerudubey@iitrpr.ac.in

ABSTRACT

With the success of crowdsourced 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 standard 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.

URL: https://github.com/descentis/kdap
Supplementary Material: https://bit.ly/2Z3tZK5

CCS CONCEPTS

• **Human-centered computing → Collaborative and social computing systems and tools**; **Wikis**; **Open source software**; *Computer supported cooperative work*; Empirical studies in collaborative and social computing.

KEYWORDS

Knowledge Building, Wikipedia, datasets, open-source library, Q&A

ACM Reference Format:
Amit Arjun Verma, S.R.S. Iyengar, Simran Setia, and Neeru Dubey. 2020. KDAP: An Open Source Toolkit to Accelerate Knowledge Building Research. In *16th International Symposium on Open Collaboration (OpenSym 2020), August 25–27, 2020, Virtual conference, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3412569.3412575

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 [26]. 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 [41]. 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 [52].

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 [12]. From predicting box office success of movies to building state-of-the-art software, these portals have helped the research communities in various aspects [13, 28, 46].

The diverse and rich content present on Wikipedia is used to study online collaboration dynamics [10, 21], to examine its impact on other online collaborative portals [45], and to train state-of-the-art artificial intelligence algorithms [30]. Similarly, the massive growth of users and posts on crowdsourced QnA portals like Stack

Overflow, Yahoo! Answers, and Quora, have attracted the attention of researchers to study the dynamics of knowledge building on these portals [9, 14, 20, 34, 36]. 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 [27, 45].

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. Based on the fact that a large fraction of researchers in the scientific community are dependent on software tools [18, 31], we aim to create an ecosystem by standardizing the representation of the datasets of crowdsourced portals and providing a toolkit to analyze these datasets. Although there is an abundance of portal-specific APIs (e.g. Wikipedia API[1], Wikidata Query Service[2], Stack Exchange API[3], Reddit API[4]), 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.

The key contributions of this paper are as follows:

(1) An XML[5]-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 present the relevant literature, including the various representation formats and tools to analyze the dataset of knowledge building portals (section 2). Then we describe the Knol-ML format and the extension mechanism associated with it (section 3). The details of the KDAP library is provided in the next section (section 4). Finally, we present the evaluation of KDAP and results of the case studies performed (section 5).

2 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.

2.1 Data Representation and Analysis

The problem of standard data representation and tools to analyze data existed 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) [7]. Further, these formats have triggered the development of open-source tools like NetworkX, iGraph, SNAP, and Gephi for analysis and visualization of graphs [1, 23]. The availability of such standard formats and analysis tools promotes open science and helps in the advancement of scientific research [32].

2.2 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.

2.2.1 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 [40], finding articles related to a word [50], getting details about edits in articles [4], 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[6], which helps user query against the wikidata[7] 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

[1]https://en.wikipedia.org/w/api.php
[2]https://query.wikidata.org/
[3]https://api.stackexchange.com/
[4]https://www.reddit.com/dev/api/
[5]https://www.w3.org/TR/REC-xml/

[6]https://query.wikidata.org/
[7]https://www.wikidata.org/wiki/Wikidata:Main_Page

are useful for sampling data from the entire set or asking specific queries, but the flexibility with data and algorithms is limited.

2.2.2 Wikipedia data parsers. Apart from web-based services, tools to extract and parse the Wikipedia data dump are available. Ferschke et al. [15] 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. [38], 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[8] data dumps (e.g. MediaWiki Utilities [16], mwxml [17]) are restricted to basic data.

2.2.3 QnA-based tools. Few QnA-based portals provide APIs for users to query the database. For instance, Stack Exchange API[9] developed by Stack Exchange network is employed to obtain various analyses results. Similarly, API provided by Reddit (Reddit API[10]) 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.

3 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 XML [6] tops the list for developing standard data representation format for crowdsourced portals. The reason being its usage in diverse areas and the availability of widely supported tools for parsing and handling XML-formatted data.

[8]https://en.wikipedia.org/wiki/Help:Wikitext
[9]https://api.stackexchange.com/docs
[10]https://www.reddit.com/dev/api/

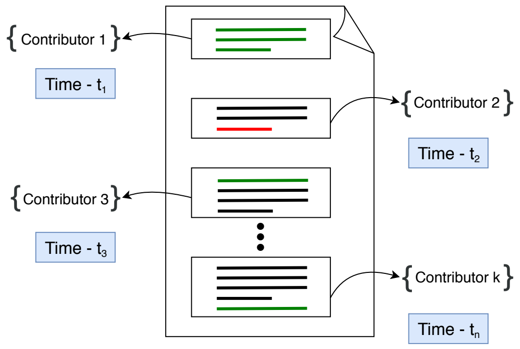


Figure 1: An illustration of *sequential knowledge* building in crowdsourced environment. A contribution can be seen in the form of addition (green) or deletion (red) information.

3.1 Core Layer

Sequential knowledge development is the fundamental feature of crowdsourced knowledge building portals, where the crowd develops knowledge sequentially by adding, deleting, and editing information [8]. 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 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 3.2 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.

```
<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>
```

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

Fig. 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:

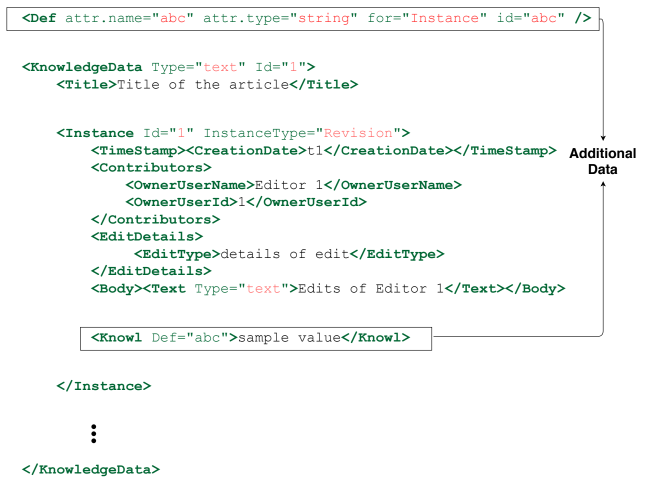
- A <TimeStamp> tag contains a sub-tag <CreationDate>, which stores the commit date and time of this edit.
- <Contributors> tag stores the details regarding editor of this instance. It has two further sub-tags, <OwnerUserName>, which stores the name of the editor and <OwnerUserId>, which stores the unique 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.

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 [47]. The full description of the data representation of wiki-based and QnA-based portals is present at: https://bit.ly/2Z3tZK5 (Appendix 1.2 and 1.3).

3.2 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. [5] have used a similar approach in



```
<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>
```

Figure 3: Representation of extension mechanism in Knol-ML.

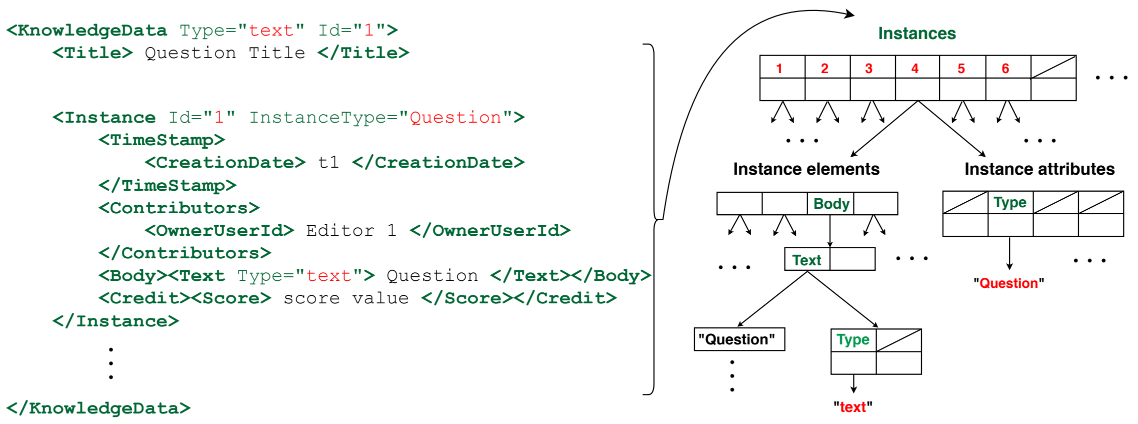
GraphML for representing additional information in graphs. They have defined the location of the additional data with the help of <key> and <data> tags. New data can be accommodated within the <data> tag, which requires a <key> providing a name and domain (type and range of values it can take) of this 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 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.

4 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



```
<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>
```

Figure 4: A diagram of knowledge data structures in KDAP. 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.

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.

4.1 Containers Functionality

KDAP provides a standard interface for both the containers, bypassing the step of pre-processing the data of multiple knowledge building portals individually. Every method in KDAP is built on the 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 dump 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.

4.2 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. Fig. 4 summarizes knowledge representations in KDAP.

Table 1: Time Complexity of Key Operations in KDAP.

Operations	Time Complexity[11]	
	Wiki-Based	**QnA-Based**
Get an instance	$O(k^2m)$	$O(n)$
Get all instances	$O(kmn)$	$O(n)$
Retrieve instance attributes	$O(1)$	$O(1)$
Add an instance	$O(1)$	$O(1)$

4.3 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

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

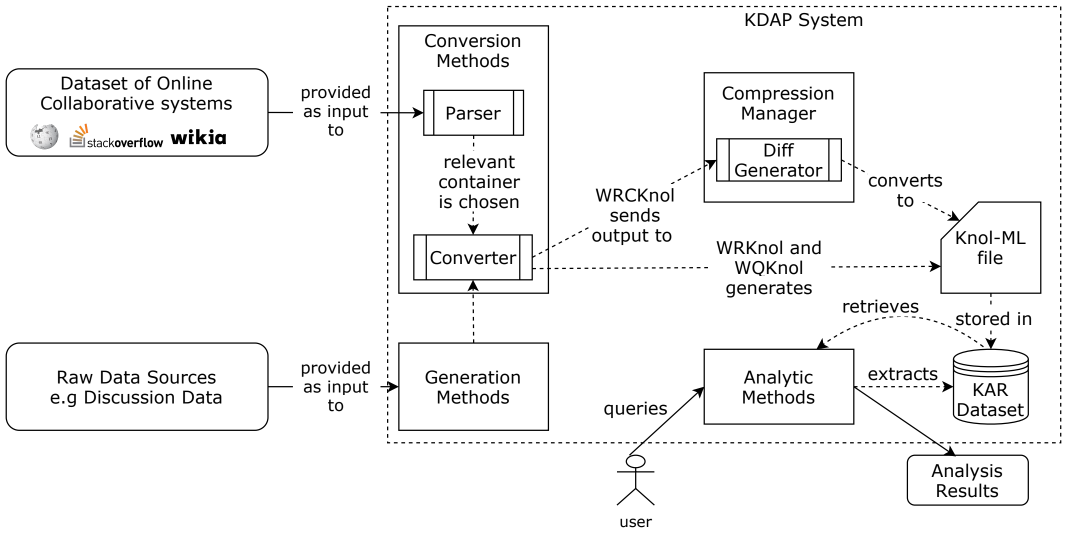


Figure 5: Overview of KDAP framework.

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 pre-requesite. Wikipedia, being a revision based system, store 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 files using the algorithm proposed by [15]. A naive algorithm will require to store the the edits made in the current revision exclusively, 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 $k \ll n$ (n is the total number of revisions). 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 $k \ll n$, 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 5), 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.

5 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 [3, 11, 22, 33], global inequality as a measure of participation [2, 44], editing patterns to understand collaboration [24, 49, 51], data extraction for various machine learning and NLP algorithms [29, 30, 42].

To evaluate our tool, we describe two experiments involving the common mining tasks as well as large-scale analyses. The tasks listed in both the experiments were performed twice, including and excluding KDAP. The authors in [39] have used a similar approach to evaluate the PyDriller tool with other existing tools. The tasks in the first experiment were designed to measure our tool based on the *ease of usage*, whereas the tasks of the second experiment evaluate the *usefulness* of our tool for large-scale analyses. We describe and compare the analysis of both the experiments. To the best of our knowledge, there does not exist any such library for the large-scale analysis of collaborative knowledge building portals. Hence, we

[11]Using the *difference* of the current revision with the previous one using the diff algorithm [19]

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 sufficient memory to hold the data in RAM. The analyses were performed 5 times, and the average execution time and memory consumption are shown.

5.1 Evaluation Based on Ease of Usage

To evaluate our tool based on the ease of usage, we compare KDAP against existing libraries and APIs for online collaborative portals. We select six commonly known knowledge building tasks that we encountered in 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 [25]), memory consumption, and execution time of both implementations. Table 3 shows the results. For all the tasks, the number of lines that are not a core functionality (for example, the constructor) is three. We keep this number as it is always the same for all the tasks for all the libraries/tools.

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.

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 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 at: https://bit.ly/2Z3tZK5 (Appendix 2), and codes for all the implementations are available in the experiment section of the GitHub repository.

5.2 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.
- 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.

5.3 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. Table 6 shows a comparison of methods implemented in KDAP with the other analysis tools.

[12] https://en.wikipedia.org/w/api.php
[13] https://effbot.org/zone/celementtree.htm
[14] https://github.com/earwig/mwparserfromhell

Table 2: Tasks assigned to the first group.

	Data Extraction
Task 1	Extracting 5 Wikipedia articles from each category namely FA, GA, B, C, Start and Stub.
Task 2	Extracting 10,000 random questions, its answers and comments from Stack Exchange site, say, anime.stackexchange.
	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.
	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.

Table 3: Comparison between KDAP and various libraries.

	Task-1		**Task-2**		**Task-3**		**Task-4**		**Task-5**		**Task-6**	
	KDAP	**(a)**	**KDAP**	**(b)**	**KDAP**	**(b & c)**	**KDAP**	**(b)**	**KDAP**	**(a)**	**KDAP**	**(b)**
Time(sec)	**7.13**	70.45	**2.91**	11.24	**521**	528	4.8	**2.7**	**80**	81.2	86	**82.3**
Memory(MB)	**0.98**	4.07	**15**	16.2	2.01	**1.9**	86	**83.2**	**0.42**	0.69	**2**	4.3
Complexity	**2**	8	**6**	13	**1**	2	**2**	4	**1**	4	**2**	4
LOC	**7**	33	**19**	88	**5**	120	**9**	36	**8**	45	**13**	70

(a), (b) and (c) refers to Wikipedia API[12], cElementTree[13] and mwparserfromhell[14] respectively

Table 4: Papers Assigned to the Participants.

Participant	Paper Assigned
P1	Black Lives Matter in Wikipedia: Collaboration and Collective Memory around OSM [43]
P2	Crowd Diversity and Performance in Wikipedia [37]
P3	An Empirical Study on Developer Interactions in StackOverflow [48]
P4	Improving Low Quality Stack Overflow Post Detection [35]

Table 5: Time and Loc Comparison for Each Participant.

Participant		With KDAP	Without KDAP	Total
P1	Time (minutes)	70	190	-63%
	LOC	19	401	-95%
P2	Time (minutes)	110	390	-71%
	LOC	50	292	-82%
P3	Time (minutes)	46	90	-48%
	LOC	34	83	-60%
P4	Time (minutes)	52	60	-13%
	LOC	81	91	-10%

6 RESOURCES

KDAP library is available at: https://kdap.github.io. The git repository contains user documentation, tutorials (with an introductory video), Knol-ML schema, stable releases of KDAP[17]. 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 Appendix for details regarding getting started with KDAP.

[17]Link for the library will be provided after acceptance of the paper

7 CONCLUSION AND FUTURE WORK

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 advance 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.

As future work, we aim to develop more functions fo the KDAP library. Also, based on the generalized structure, visualization methods can be implemented to visualize the knowledge data. KDAP system can also be implemented in the C programming language, optimizing the overall running time and memory usage. We encourage readers to install KDAP and explore its functionalities,

Table 6: Comparison of KDAP methods with other analysis tools.

Methods	KDAP	WikiBrain	JWPL[15]	DBpedia[16]	Wikipedia API	SE API
Extraction Methods						
Wikipedia article extraction by name	Yes	Yes	Yes	No	🟨 Yes	No
Wikipedia article extraction by class	Yes	No	No	No	🟨 Yes	No
Stack Exchange extraction by portal name	Yes	No	No	No	No	🟨 Yes
Pageviews extraction of Wikipedia/Stack Exchange	Yes	🟨 Yes	No	No	🟨 Yes	🟨 Yes
Parsing Methods						
Link/Category/Image extraction	Yes	Yes	Yes	Yes	🟨 Yes	🟨 Yes
Revision/text extraction	Yes	Yes	Yes	Yes	🟨 Yes	No
Contributors' details extraction	Yes	No	No	No	🟨 Yes	🟨 Yes
Question/Answers/Comments extraction	Yes	No	No	No	No	🟨 Yes
Knowledge Building Methods						
Semantic Relatedness	No	Yes	No	Yes	No	No
Edit Statistics	Yes	No	No	No	No	No
Measure of Inequality	Yes	No	No	No	No	No
Controversy detection	Yes	No	No	No	No	No

🟨 Yes represents that the number of calls are limited.

most importantly, we encourage researchers and developers to help improve KDAP by joining our team on GitHub.

ACKNOWLEDGMENTS

This work was funded by the assistance received from CSRI, Department of Science and Technology India via grant no. SR/CSRI/344/2016

REFERENCES

[1] Nadeem Akhtar. 2014. Social network analysis tools. In *2014 Fourth International Conference on Communication Systems and Network Technologies*. IEEE, 388–392.

[2] Ofer Arazy and Oded Nov. 2010. Determinants of wikipedia quality: the roles of global and local contribution inequality. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM, 233–236.

[3] Joshua E Blumenstock. 2008. Size matters: word count as a measure of quality on wikipedia. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 1095–1096.

[4] Nadia Boukhelifa, Fanny Chevalier, and Jean-Daniel Fekete. 2010. Real-time aggregation of wikipedia data for visual analytics. In *2010 IEEE Symposium on Visual Analytics Science and Technology*. IEEE, 147–154.

[5] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. 2001. GraphML progress report structural layer proposal. In *International Symposium on Graph Drawing*. Springer, 501–512.

[6] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and Franois Yergeau. 2000. Extensible markup language (XML) 1.0.

[7] Ken Cherven. 2015. *Mastering Gephi network visualization*. Packt Publishing Ltd.

[8] Anamika Chhabra and SRS Iyengar. 2017. How Does Knowledge Come By? *arXiv preprint arXiv:1705.06946* (2017).

[9] Anamika Chhabra and SRS Iyengar. 2020. Activity-selection Behavior of Users in StackExchange Websites. In *Companion Proceedings of the Web Conference 2020*. 105–106.

[10] Anamika Chhabra and SR Sudarshan Iyengar. 2018. Characterizing the Triggering Phenomenon in Wikipedia. In *Proceedings of the 14th International Symposium on Open Collaboration*. 1–7.

[11] Denzil Correa and Ashish Sureka. 2014. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In *Proceedings of the 23rd international conference on World wide web*. ACM, 631–642.

[12] Ulrike Cress and Joachim Kimmerle. 2008. A systemic and cognitive view on collaborative knowledge building with wikis. *International Journal of Computer-Supported Collaborative Learning* 3, 2 (2008), 105.

[13] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, 1277–1286.

[14] Zhenhua Dong, Chuan Shi, Shilad Sen, Loren Terveen, and John Riedl. 2012. War versus inspirational in forrest gump: Cultural effects in tagging communities. In *Sixth International AAAI Conference on Weblogs and Social Media*.

[15] Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. 2011. 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*. Association for Computational Linguistics, 97–102.

[16] Aaron Halfaker. 2019. *MediaWiki Utilities*. Retrieved February 28, 2019 from https://pythonhosted.org/mediawiki-utilities/

[17] Aaron Halfaker. 2019. *MediaWiki XML Processing*. Retrieved March 21, 2019 from https://pythonhosted.org/mwxml/

[18] James Howison and Julia Bullard. 2016. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology* 67, 9 (2016), 2137–2155.

[19] James Wayne Hunt. [n.d.]. *An algorithm for differential file comparison*.

[20] Imrul Kayes, Nicolas Kourtellis, Daniele Quercia, Adriana Iamnitchi, and Francesco Bonchi. 2015. Cultures in community question answering. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*. ACM, 175–184.

[21] Aniket Kittur and Robert E Kraut. 2010. Beyond Wikipedia: coordination and conflict in online production groups. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM, 215–224.

[22] Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 591–602.

[23] Jure Leskovec and Rok Sosič. 2016. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1.

[24] Jun Liu and Sudha Ram. 2011. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Transactions on Management Information Systems (TMIS)* 2, 2 (2011), 11.

[25] Thomas J McCabe. 1976. A complexity measure. *IEEE Transactions on software Engineering* 4 (1976), 308–320.

[26] Erin C McKiernan, Philip E Bourne, C Titus Brown, Stuart Buck, Amye Kenall, Jennifer Lin, Damon McDougall, Brian A Nosek, Karthik Ram, Courtney K Soderberg, et al. 2016. Point of view: How open science helps researchers succeed. *Elife* 5 (2016), e16800.

[27] Connor McMahon, Isaac Johnson, and Brent Hecht. 2017. The substantial interdependence of wikipedia and google: A case study on the relationship between peer production communities and information technologies. In *Eleventh International AAAI Conference on Web and Social Media*.

[28] Márton Mestyán, Taha Yasseri, and János Kertész. 2013. Early prediction of movie box office success based on Wikipedia activity big data. *PloS one* 8, 8 (2013), e71226.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.
[31] Udit Nangia, Daniel S Katz, et al. 2017. 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).*
[32] Brian A Nosek, George Alter, George C Banks, Denny Borsboom, Sara D Bowman, Steven J Breckler, Stuart Buck, Christopher D Chambers, Gilbert Chin, Garret Christensen, et al. 2015. Promoting an open research culture. *Science* 348, 6242 (2015), 1422–1425.
[33] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2014. Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th international workshop on social software engineering.* ACM, 33–36.
[34] Nigini Oliveira, Michael Muller, Nazareno Andrade, and Katharina Reinecke. 2018. The Exchange in StackExchange: Divergences between Stack Overflow and its Culturally Diverse Participants. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 130.
[35] Luca Ponzanelli, Andrea Mocci, Alberto Bacchelli, Michele Lanza, and David Fullerton. 2014. Improving low quality stack overflow post detection. In *2014 IEEE International Conference on Software Maintenance and Evolution.* IEEE, 541–544.
[36] Giovanni Quattrone, Afra Mashhadi, and Licia Capra. 2014. 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.* ACM, 934–944.
[37] Ruqin Ren and Bei Yan. 2017. 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.* ACM, 6342–6351.
[38] Shilad Sen, Toby Jia-Jun Li, WikiBrain Team, and Brent Hecht. 2014. WikiBrain: democratizing computation on Wikipedia. In *Proceedings of The International Symposium on Open Collaboration.* ACM, 27.
[39] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. 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.* ACM, 908–911.
[40] Thomas Steiner, Seth Van Hooland, and Ed Summers. 2013. 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.* ACM, 791–794.
[41] Harry Thornburg. 2014. *Four main languages for Analytics, Data Mining, Data Science.* Retrieved March 14, 2019 from https://www.kdnuggets.com/2014/08/four-main-languagesanalytics-data-mining-data-science.html.
[42] Christoph Treude and Martin P Robillard. 2016. Augmenting api documentation with insights from stack overflow. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE).* IEEE, 392–403.
[43] Marlon Twyman, Brian C Keegan, and Aaron Shaw. 2017. 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.* ACM, 1400–1412.
[44] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. 2013. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *2013 International Conference on Social Computing.* IEEE, 188–195.
[45] Nicholas Vincent, Isaac Johnson, and Brent Hecht. 2018. 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.* ACM, 566.
[46] Gang Wang, Konark Gill, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. 2013. Wisdom in the social crowd: an analysis of quora. In *Proceedings of the 22nd international conference on World Wide Web.* ACM, 1341–1352.
[47] G Alan Wang, Harry Jiannan Wang, Jiexun Li, Alan S Abrahams, and Weiguo Fan. 2015. An analytical framework for understanding knowledge-sharing processes in online Q&A communities. *ACM Transactions on Management Information Systems (TMIS)* 5, 4 (2015), 18.
[48] Shaowei Wang, David Lo, and Lingxiao Jiang. 2013. An empirical study on developer interactions in StackOverflow. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing.* ACM, 1019–1024.
[49] Adam Wierzbicki, Piotr Turek, and Radoslaw Nielek. 2010. Learning about team collaboration from Wikipedia edit history. In *Proceedings of the 6th international symposium on Wikis and open collaboration.* ACM, 27.
[50] Zhaohui Wu and C Lee Giles. 2015. Sense-aaware semantic analysis: A multi-prototype word representation model using wikipedia. In *Twenty-Ninth AAAI Conference on Artificial Intelligence.*
[51] Jie Yang, Claudia Hauff, Alessandro Bozzon, and Geert-Jan Houben. 2014. Asking the right question in collaborative q&a systems. In *Proceedings of the 25th ACM conference on Hypertext and social media.* ACM, 179–189.
[52] Rongying Zhao and Mingkun Wei. 2017. Impact evaluation of open source software: An altmetrics perspective. *Scientometrics* 110, 2 (2017), 1017–1033.

A GETTING STARTED WITH KDAP

A.1 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 present at: https://bit.ly/2Z3tZK5 (section 2) for more information.

A.2 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)
```

A.3 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)
```

A.4 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
```

```python
    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
```

A.5 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:

```python
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'])
```