1 Appendix

1.1 Knol-ML: Core Layer

```
<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 1. Representation of sequential knowledge data in Knol-ML format.

Figure 1 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.

1.2 Wiki Based Data

Under the umbrella of *Wikimedia Foundation*, Wikipedia[1] is a free encyclopedia, where anyone can edit and contribute knowledge. As of March 2019, English Wikipedia contains

[1]https://www.wikipedia.org/

```
<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>
    </Instance>
              ⋮
</KnowledgeData>
```

Figure 2. Representation of Wikipedia data in Knol-ML.

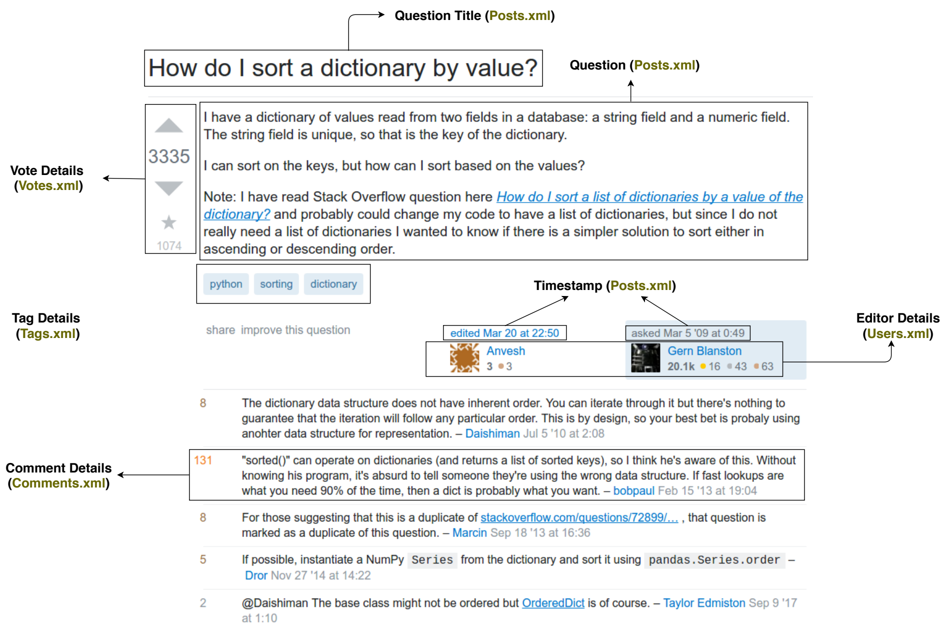
5,817,607 articles. Wikimedia foundation release data dumps of Wikipedia and all of its projects on a regular basis. The data dump comprises of the following:

- Text of the current and all the previous revisions as XML files with defined wrapper format[2].
- Information about other Wikipedia pages like *Talk*, *User Talk*, *Templates*, etc.

Our focus is on Wikipedia articles, which are the main source of the referred knowledge. The XML dump of articles contains information like, *title* and edits in the form of *revisions*. Each revision has details like, *date and time* of the edit, *editors' information*, *comments*, if any and the actual *edits* in text form. For every such detail, a tag has been defined. The XML dump may contain multiple pages, each representing a different article. Each article is considered as `<KnowledgeData>` in Knol-ML format. Figure 2 depicts the Knol-ML format for Wikipedia data. The details of the tags are as follows:

- `<Title>` stores the title of the article.
- `<Instance>` stores all the information related to a particular edit, containing the following tags:
  - `<TimeStamp>` stores the date and time of the edit.
  - `<Contributors>` stores the editors' information using the subtags `<OwnerUserId>` and `<OwnerUserName>`.
  - `<EditDetails>` stores the comment related to the edit.
  - `<Body>` stores the actual edit in the `<Text>` subtag.

[2]https://www.mediawiki.org/xml/export-0.8.xsd



How do I sort a dictionary by value?

I have a dictionary of values read from two fields in a database: a string field and a numeric field. The string field is unique, so that is the key of the dictionary.

I can sort on the keys, but how can I sort based on the values?

Note: I have read Stack Overflow question here *How do I sort a list of dictionaries by a value of the dictionary?* and probably could change my code to have a list of dictionaries, but since I do not really need a list of dictionaries I wanted to know if there is a simpler solution to sort either in ascending or descending order.

8 The dictionary data structure does not have inherent order. You can iterate through it but there's nothing to guarantee that the iteration will follow any particular order. This is by design, so your best bet is probaly using anohter data structure for representation. – Daishiman Jul 5 '10 at 2:08

131 "sorted()" can operate on dictionaries (and returns a list of sorted keys), so I think he's aware of this. Without knowing his program, it's absurd to tell someone they're using the wrong data structure. If fast lookups are what you need 90% of the time, then a dict is probably what you want. – bobpaul Feb 15 '13 at 19:04

8 For those suggesting that this is a duplicate of stackoverflow.com/questions/72899/... , that question is marked as a duplicate of this question. – Marcin Sep 18 '13 at 16:36

5 If possible, instantiate a NumPy `Series` from the dictionary and sort it using `pandas.Series.order` – Dror Nov 27 '14 at 14:22

2 @Daishiman The base class might not be ordered but OrderedDict is of course. – Taylor Edmiston Sep 9 '17 at 1:10

Figure 3. An illustration of Stack Overflow data storage.

1.3 QnA Based Data

Unlike wiki-based portals, knowledge building in QnA-based portals like, Stack overflow[3] and Quora[4], develops through a series of discussions on a particular topic. The discussions may involve series of questions and answers, but not necessarily. The highlight of these portals is the explicitness of knowledge as the crowd provides question specific answers.

Portals like *Stack Overflow* and *Ask Ubuntu* are part of *Stack Exchange Network*. Currently, there are 185 QnA based portals maintained by Stack Exchange. Domain of these portals varies from programming to music. Similar to Wikipedia, Stack Exchange provides data dump of its websites in XML format. Data dump of each portal contains eight XML files, namely, *Posts.xml*, *PostHistory.xml*, *Users.xml*, *Comments.xml*, *Tags.xml*, *Votes.xml*, *Badges.xml* and *PostLinks.xml*, each of which contains different set of information. For example, *Posts.xml* contains all the questions and answers. *Users.xml* contains information related to all the users, *Comments.xml* contains comments of posts and related information.

Based on the design of QnA portal, every question sequentially accumulates posts like answers, comments, votes, etc. Our aim is to combine the data of these portals into sequential knowledge building format. A *thread* is defined as a question and all the posts related to it (answers, comments, votes, scores, etc.) [?]. We represent the information of each thread in the form of knowledge data. A question, an answer or a comment associated with a time-stamp is considered as an instance of Knol-ML. For example, a question q_1 may arrive at time t_1, followed by an answer a_1 at time t_2 ($t_i < t_{i+1}$). The answer a_1 may get multiple comments, hence, a post (question, answer or comment) arriving at time-stamp

[3]https://stackoverflow.com/
[4]https://www.quora.com/

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


    <Instance Id="1" InstanceType="Question">
        <TimeStamp>
            <CreationDate>t1</CreationDate>
            <LastActivityDate>t2</LastActivityDate>
        </TimeStamp>
        <Contributors>
            <OwnerUserName>Editor 1</OwnerUserName>
            <ownerUserId>1</OwnerUserId>
            <LastEditorUserName>Editor 3</LastEditorUserName>
            <LastEditorUserId>3</LastEditorUserId>
        </Contributors>
        <Body><Text Type="text">Question by Editor 1</Text></Body>
        <Tags>tags</Tags>
        <Credit><Score>score value</Score></Credit>
    </Instance>


    <Instance Id="2" InstanceType="Comment">
        <TimeStamp>
            <CreationDate>t3</CreationDate>
        </TimeStamp>
        <Contributors>
            <OwnerUserName>Editor 4</OwnerUserName>
            <OwnerUserId>4</OwnerUserId>
        </Contributors>
        <Body><Text Type="text">Comment by Editor 4</Text></Body>
        <Credit><Score>score value</Score></Credit>
    </Instance>
</KnowledgeData>
```

Figure 4. Representation of Stack Overflow data in Knol-ML.

t_i will be considered as i^{th} instance. There could be multiple information associated with each instance like *upvotes*, *downvotes*, *editor's detail*, etc. All such information will be represented within the corresponding instance.

Figure 4 is an example representing a Stack Overflow thread. A single Knol-ML file can have multiple threads, each described within `<KnowledgeData>` tag. The tags `<Tag>` and `<Credit>` store the category of the topic discussed in a thread and reputation for a post, respectively. Extra information like, editor's biography, badge details, etc., can be represented using the extension mechanism. All the QnA based knowledge building portals maintain the basic structure of threaded questions and answers, making it feasible to represent in Knol-ML format.

2 Evaluation Based on Ease of Usage

2.1 Data Extraction

All the knowledge building analyses require one to extract the relevant dataset. Proper extraction tools are not available for mining the dataset of collaborative knowledge building portals. For example, as mentioned by Milne et al. [?], a significant amount of effort is required to mine and preprocess the dataset of Wikipedia. With KDAP, a user can easily retrieve wiki-based and QnA-based datasets using the corresponding container functions. We have implemented retrieval algorithms for Wikipedia and Stack Exchange dataset. Future version of KDAP will include the mining functions of various other knowledge building portals. We now explain some of the mining methods for Wikipedia and Stack Exchange network.

2.1.1 Wikipedia Data Extraction

Even though Wikipedia dataset is open, non-trivial amount of work is required to extract and analyse it. KDAP facilitates users with extraction methods which are easy to use and are powerful enough to query complex datasets. To perform a large-scale analysis of Wikipedia data, it is necessary to extract the complete information related to a query. For instance, Ren et al. [?] created their dataset of Wikipedia articles sampled from nine Wikiprojects namely, History, Sociology, Geography, Culture, Technology, Mathematics, Science, Philosophy, and Religion. As another example, Zhang et al. [?] have used articles which belong to *Good Articles* and *Featured Articles* category. KDAP simplifies the extraction process by providing direct methods. For example `download_dataset` method can be used to retrieve the articles' full revision history based on a category. To replicate the dataset created by Ren et al. for their analysis, following lines of code can be used:

```
import kdap
knol = kdap.knol()
wikiproject_list = ['History', 'Scociology', 'Geography', 'Culture', '
    Technology', 'Science', 'Mathematics', 'Philosophy', 'Religion']
article_list = []
for project in wikiproject_list:
    article_list += knol.get_wiki_article_by_class(wikiproject=project)
knol.download_dataset(sitename='wikipedia', article_list=article_list)
```

The above code will query the database which has been created as a part of KDAP library. The database contains entries related to an article like category, class and the Wikipedia dump it belongs to. The algorithm chooses the relevant methods to extract the dataset. For example, if the queried articles are not available in the KAR dataset then relevant Wikipedia dumps are used to extract the articles and store them in Knol-ML format. Similarly, `get_articles_by_class` method can be used to extract the articles based on a class, where class name is passed as an argument. The KDAP documentation contains the list of extraction methods and related details.

2.1.2 Stack Exchange Data Extraction

Similar to Wikipedia, Stack Exchange also provides data dump for each of its portals. The dataset can be manually downloaded and analysed. KDAP provides complex query methods for Stack Exchange dataset. Similar to Wikipedia, `download_dataset` method can be used to retrieve the dataset of a Stack Exchange portal. Portal name can be passed as an argument to this function.

2.2 Edit Statistics

Researchers have used edit related statistics to perform various tasks ranging from studying the gender imbalance on Wikipedia [?] to designing machine learning models to predict the sentiment of answers that a particular question will attract [?]. These statistics play an important role in understanding the knowledge building dynamics. For example, Elia [?] defined the readability of Wikipedia article based on various indices like word length, sentence length, and lexical density and found that there is no significant difference between

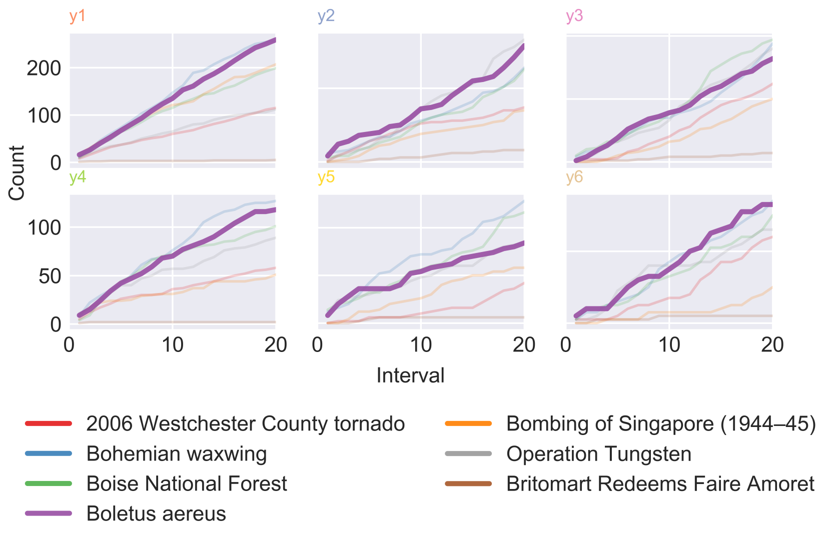


Figure 5. The figure depicts a few language statistics. *y1, y2, y3, y4, y5* and *y6* represents the count of *Content Added*, *Content Reorganized*, *Content Deleted*, *Hyperlinks Added*, *Hyperlinks Fixed*, and *Hyperlinks Deleted* throughout the timeline, respectively. For comparison purpose, the timeline of each article has been divided into 20 equal intervals and edit statistics are calculated at these intervals. Thickness of the line for *Boletus aereus* article is to highlight the comparison with other articles on the aforementioned parameters.

Wikipedia and Britannica Encyclopedia based on these parameters. Blumenstock [?] has also used word count to show that it is positively correlated with the article quality on Wikipedia. On similar lines, Ponzanelli et al. [?] used various edit statistics to identify low-quality posts in Stack Overflow.

As a part of KDAP, we have implemented edit-based analysis methods. `get_edit_stats` is one such method which can be used to obtain content-related statistics. Following lines of code can be used to get these edit statistics:

```
import kdap
knol = kdap.knol()
edit_stats = knol.get_edit_stats(dir_path='directory_path_of_data', slab
    =20)
```

Figure 5 represents the output of above code on seven Wikipedia articles.

2.3 Measure of Inequality

There are various ways to measure the inequality of contribution among users of a portal. A common measure is Gini Coefficient [?] which is used to measure the inequality of a distribution in the fields of economics and sociology, for example, distribution of wealth in a country. The value of Gini coefficient ranges from 0 to 1; a value close to 0 denotes that the contribution is equally distributed amongst contributors, whereas a coefficient close to 1 denotes that most of the contribution is by a small group of contributors.

Table 1. Execution Time and Memory Requirements for calculating Local Gini Coefficient for various portals.

Portal	Time (seconds)		Memory (kilo bytes)	
	Traditional*	KDAP	Traditional*	KDAP
ai.stackexchange	4.25	0.45	2015	1600
anime.stackexchnage	253	1.62	14736	6012
astronomy.stackexchange	79.2	1.88	4296	4240
stephenking.wikia	21	10.48	13788	3200
valkyria.wikia	2.89	1.03	1702	2608

* refer to section section 2.3

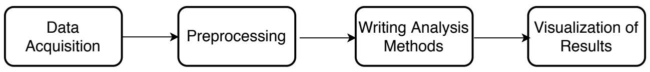


Figure 6. Steps required to measure the Gini Coefficient.

It has been observed that highly active collaborative portals have a higher Gini coefficient as compared to lesser active portals [**? ?**]. To state the result that South American countries tend to have a higher percentage of contributing users in Stack Overflow and Superuser, as compared to the users of other countries [**?**], authors had to perform a series of steps listed in Figure 6 to measure the Gini coefficient of these portals. Also, *traditional* algorithm requires the parser to retrieve the text through two different files (*Post.xml* and *Comments.xml*) to calculate the overall contribution of a user.

We have implemented the algorithm to calculate both Local and Global Gini coefficient, taking care of time and memory consumption. Local Gini coefficient is calculated for a knowledge data which may represent an article of wiki-based data or a thread of QnA-based data. Similarly, Global Gini coefficient is calculated for a portal, considering the contributions of all users throughout the portal. Each `<Instance>` tag in a Knol-ML file contains an attribute named `Bytes` which stores the size of the contribution of that instance. The KDAP method for calculating the Gini coefficient makes use of this information, avoiding the extra time and memory required to extract and store the text of that instance respectively. Table 1 represents the execution time and memory requirement for calculating Local Gini coefficient.

A comparison of knowledge building portals based on Global Gini coefficient which will require the steps listed in Figure 6 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='
```

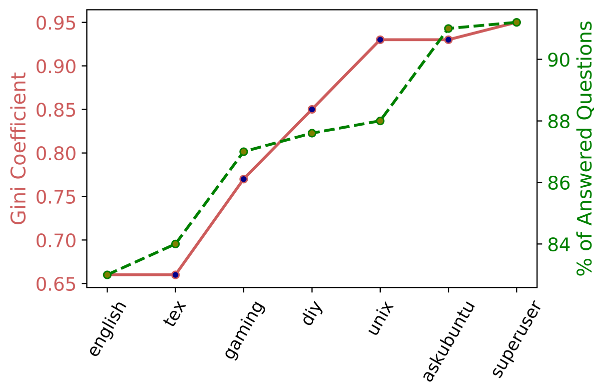


Figure 7. Comparison of Gini coefficient with percentage of answered questions on Stack Exchange portals.

```
    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'])
```

Figure 7 illustrates the comparison of Global Gini coefficient of seven QnA portals with the respective percentage of answered questions. As shown in Table 1, KDAP method for calculating Gini coefficient outperforms the traditional way (described in Figure 6) in terms of both time and memory. With fewer lines of code, complex algorithms can be executed across portals. As a part of the inequality measure, we have also implemented methods which can be found in the documentation[5].

[5]http://kdap.github.io