Wikipedia Articles Quality Evaluation Using Hubs and Authorities

Tianhao Li tl1924@nyu.edu

Tandon School of Engineering, New York University

1. Introduction

Wikipedia is the largest and most popular general reference work on the Internet [1] and is ranked among the ten most popular websites. As of February 2014, it had 18 billion page views and nearly 500 million unique visitors each month. Wikipedia pages each have their own history with detailed information about every revision, and this public log is one of the core aspects of the project [2].

At the meantime, Wikipedia, as an encyclopedia evolved in wiki approach, can be edited by any contributors from all over the world and in different expertise levels on topics. As a result, The Wikipedia articles can be very varying in terms of quality.

Is there any relation between an articles' edit history and its quality? Intuitively, good-quality articles might have been edited by many contributors many times, especially good contributors. Reversely, a good contributor must have written many articles with good quality. High-quality articles and good contributors define each other mutually and recursively. Thus, I think HITS (Hyperlink-Induced Topic Search) algorithm is a good fit to this scenario.

2. Algorithm

HITS algorithm is originally to rank query results in web search engine. In HITS algorithm, given a query, every web page is assigned two scores. One is called its hub score and the other its authority score. For any query, we compute two ranked lists of results rather than one. The ranking of one list is induced by the hub scores and that of the other by the authority scores. A good hub page is one that points to many good authorities; a good authority page is one that is pointed to by many good hub pages. We thus appear to have a circular definition of hubs and authorities [8].

In our scenario, I modified the algorithm in order to fit our research purpose. I used all articles as the hubs side, while all contributors list as the authority side.

And the link between them is the edits made on one article by one contributor.

For the articles in the dataset, I use $\mathrm{h(u)}$ to denote the scores of each hub. For contributors, I use $\mathrm{a(v)}$ to denote scores of each contributors. Initially, I set all hub scores and authority scores to 1. If contributor v has edited article u, then $\mathrm{v} \rightarrow \mathrm{u}$ and $\mathrm{u} \rightarrow \mathrm{v}$ exists. The core of HITS algorithm is a pair of updates plus a normalization function to the hub scores and authority scores.

$$h(u) \leftarrow \sum_{v \to u} a(v)$$

$$a(u) \leftarrow \sum_{u \to v} h(v)$$

The normalization function would simply normalize the summation of hub scores or authority scores to 1. Based on HITS algorithm [8], after a number of iterations, all of these scores would converge, and the rank of hubs and authorities would be stable.

3. Dataset

Wikimedia Foundation releases data dumps of Wikipedia on a regular basis. English Wikipedia is dumped once a month [3]. It includes text and metadata of current or all revisions of all pages as XML files in compressed format. The data is dumped semi-monthly. As of 22 December 2016, there are 5,313,354 articles in the English Wikipedia, the data volume would be huge if we use full revision history data including edit text. Hence, I decided to use only the metadata of revision history.

For dump of November 20, 2016, there are about 300 gigabytes data after decompressed. Each file includes a bunch of articles and their revision history from the very beginning of this article. The data is in XML format.

For example:

```
<page>
<title>AccessibleComputing</title>
   <ns>0</ns>
   <id>10</id>
   <redirect title="Computer
accessibility" />
   <revision>
     <id>233192</id>
     <timestamp>2001-01-
21T02:12:21Z</timestamp>
     <contributor>
    <username>RoseParks</username>
       <id>99</id>
     </contributor>
     <comment>*</comment>
     <model>wikitext</model>
     <format>text/x-wiki</format>
     <text id="233192" bytes="124" />
     <sha1>8kul9tlwjm9oxgvqzbwuegt9b
2830vw</sha1>
   </revision>
    <revision>
      …
    </revision>
  </page>
  <page>
    …
  </page>
```

Above is an example of revision history. Under each page, there are several attributes. Here I only list attributed that I used.

<page>: including information about an article and its revision history
   <title>: the title of this article
   <ns>: namespace, A Wikipedia namespace is a set of Wikipedia pages whose names begin with a particular reserved word (followed by colon) [7].
   <id>: id of this article
    <redirect>: a redirect is a page which automatically sends visitors to another page, usually an article or section of an article. The attribute "title" is the title of the article it redirects to.
    <revision>: a record of revision metadata. All revision are sorted chronically.
        <id>: id of this revision
        <timestamp>: the time this revision was made
        <contributor>: including information about the contributor made this edit.
            <username> and <id>: only appear if user edited as logged in
            <ip>: only appears if user edited but not logged in
        <minor/>: indicates there are only superficial differences exist between the current and previous versions.

For namespace, Wikipedia has 34 current namespaces: 16 subject namespaces, 16 corresponding talk namespaces, and 2 virtual namespaces. For example, in the user namespace all titles begin with the prefix User:. In the case of the article (or main) namespace, in which encyclopedia articles appear, the reserved word and colon are absent [7].

For minor edits, examples include typographical corrections, formatting and presentational changes, and rearrangements of text without modification of its content. A *minor edit* is one that the editor believes requires no review and could never be the subject of a dispute [6]. Any change that affects the *meaning* of an article is not minor, even if it concerns a single word [6].

4. Data parsing and cleaning

4.1 Data parsing

As discussed above, the data dump is in XML form in 300 GB volume. We need to parse the data to run our algorithm. According to the algorithm we discussed above, we need all articles as hubs, all contributors as authorities, and edits as the links(edges) between articles and contributors.

Articles are uniquely denoted by their id, and also have a title which might not be unique. Logged-in contributors also have unique id for each of them. They may have multiple usernames as they are allowed to change their usernames at any time. Non-logged-in contributors have no unique id and username. Their IPs are unique though, and can be used as id.

As a result, we can use unique id to represent articles and contributors in our program, which would make our program more efficient. Hence, my parser should parse the raw data into three parts. Pages, contributors and revisions.

Pages includes all articles in the form of
article-id, article-title (, article-redirect).

Contributors includes all contributors in the form of
contributor-id, contributor-username1, contributor-id/IP2…
or only
contributor-IP.

Revisions includes all edit as a record of

article-id, contributor-id/IP1, contributor-id/IP2, contributor-id/IP3, contributor-id/IP4…

4.2 Data cleaning

When parsing the data, the parser should also deal with the noise in the data. The noises are discussed as following.

4.2.1 Unwanted namespace

As mentioned in the data description, there are 34 current namespace covering all pages in Wikipedia. But we only interest in main namespace which contains all encyclopedia articles, lists, disambiguation pages, and encyclopedia redirects [10].

4.2.2 Bots' edits

A bot is an automated or semi-automated tool that carries out repetitive and mundane tasks [9]. Bots' edits may not improve the content quality of articles at all but their might edits a large amount of times, which maybe promote their authorities score.

However, this is unwanted. We thus would filter all edits made by bots. There is a list of one thousand registered bots in Wikipedia [10]. If any of the contributors' username appears in this list, the parser would discard that record.

4.2.3 Continuous edits.

Users usually have the habits of saving intermediate revisions to avoid loss of work due to unexpected hardware, software or network errors. By removing these intermediate revisions, we reduced the computation time without losing necessary interaction data [11].

4.2.4 Minor edits

As described above, minor edits only fix language issues, and will not change the meaning of content. Intuitively, these edits also would not improve the quality of the articles, neither would gain any reputation for these contributors.

After parsing and cleaning the data we have, I got the around 3 GB revisions stored in separate files.

5. Implementation Details

5.1 Parser

The parser is implemented in Java with *SAXPaser* library. The parser includes five classes, *Parser, Page, Contributor, Revision, RevisionList.*
When parsing a data file, the parser would loop through each line of the file. If the Parser encountered a new page, it would create a page object and a revisionList object, and wrote only the information about this page to *Pages* file mentioned above. After, if the Parser encountered a revision of this page, a revision object would be created and added to the revisionlist object of this page. At the end of each page, the Parser would write the page and revision list of this page to disk.

In each revision, the Parser would construct a contributor instance when encountered. Also the Contributor class would maintain a *hashmap* to record all distinct contributors encountered so far in this file. At the end of parsing this file. This *hashmap* would be dumped to disk as *Pages.* This result in a local distinct contributors list. Since there are totally 27 data files of November 20, 2016 dump, for each files, we would have a list of distinct contributors in that file. However, there could be a large overlap of contributors lists over all files. Hence, when implement HITS algorithm and read the contributors lists from all 27 files, there should be a function to fillter out the duplicated contributors across these files.

5.2 HITS on Spark(Scala)

As you can see, the revision data volume is still huge after filter out unwanted noises, plus around 1.5 GB *contributors* data and around 1 GB *articles* data. Especially for algorithm like HITS, it requires several iterations over the whole graph of contributors between articles. This is very heavy. I thus decided to use Spark. Spark has the capability to process iterative algorithms like HITS.

I implemented the HITS algorithm in Scala. Spark is implemented in Scala, and it also provides concise and light weighted APIs to Scala users.

In the Scala application, program would read the Pages, and transform it to a RDD (Resilient Distributed Dataset) in the schema of

article-id, article-name.

For *contributors* files, it would read it to a RDD in the schema of

contributor-id, username.

After read all contributors, Spark provides a nice API to reduce the duplicate contributor in RDD. This would reduce the memory consume.

For *revisions*, the original format is

article-id, contributor-id/IP1, contributor-id/IP2, contributor-id/IP3, contributor-id/IP4…

But the HITS algorithm requires links/edges between articles and contributors. Hence, the application need to parse it to the form of

article-id1, contributor-id/IP1
article-id1, contributor-id/IP2
article-id1, contributor-id/IP3
……
article-id2, contributor-id/IP1
article-id2, contributor-id/IP2
article-id2, contributor-id/IP3
……

The implementation of the updates and normalization functions are exactly same as described in the algorithm.

I parsed the data locally, and uploaded the intermediate data files into AWS S3. Then run the Scala on *Databricks'* Spark cloud platform to get the results. The result are files including the articles sorted by their hubs scores in descending order.

6. Evaluation of results

6.1 WikiProjects Assessment

In 2007, in preparation for producing a print version, the English Wikipedia introduced an assessment scale of the quality of articles. Articles are rated by WikiProjects. The range of quality classes begins with "Stub" (very short pages), followed by "Start", "C" and "B" (in increasing order of quality) [5].

The global summary table below is computed by taking the highest quality and importance rating for each assessed article in the main namespace [5].

Table 1: All rated articles by quality. *[5]*

Quality	Total
Featured Articles(FA)	5,893
Featured Lists(FL)	2,068
A-class Articles(A)	1,660
Good-Articles(GA)	27,692
B-class Articles(B)	110,293
C-class Articles(C)	236,252
Start	1,453,545
Stub	2,923,925
List	199,759
Assessed	**4,960,887**
Unassessed	**536,472**
Total	**5,497,359**

This assessment made by WikiProject would be the test base of my rank. The FA/FL class is the best part of Wikipedia articles. The rest of the quality labels are in descending order. i.e. FA/FL > A > GA > B > Start > Stub. The Start and Stub class articles only contain a little content.

6.2 Evaluation Model

We only care about have those good articles been ranked highly in the results. We would only interested in top subset of the returns.

In the context of information retrieval, precision and recall are two measures unranked retrieval sets. Even though the HITS algorithm would return ranked results based on hub or authority scores, our test truth base is not ranked. Hence, we consider the top of HITS result also as a set.

Precision (P) is the fraction of retrieved documents that are relevant .*Recall (R)* is the fraction of relevant documents that are retrieved [8].

For simplicity, in our case, we only consider the set of FA articles. Featured articles are considered to be the best articles Wikipedia has to offer, as determined by Wikipedia's editors. They are used by editors as examples for writing other articles. Before being listed here, articles are reviewed as featured article candidates for accuracy, neutrality, completeness, and style according to our featured article criteria [5].

There are about 5000 articles are Featured Articles in WikiProjects Assessment that I can get easily.

We define:

Precision(P) as the fraction of returned top-5000 articles that are Featured Articles,

Recall(R) as the fraction of Featured Articles that are returned top-5000 articles.

	FA	Non-FA
Top-5000	True positives(tp)	False positives(fp)
Not top-5000	False negatives(fn)	True negatives(tn)

Then:

$$P = tp/(tp + fp)$$
$$R = tp/(tp + fn)$$

6.3 Evaluation on results

Wikipedia provided the list of all Featured Articles. I parsed it in to articles' titles list and got 5061 articles. Diff with my results and got the table below.

	FA	Non-FA
Top-5000	2356	2644
Not top-5000	2705	

Thus, P = 47.12%, R = 46.75%. We can see that the recall and precision is not very high. I think this is because I used a very naïve model to implement HITS. There a lot of features can be added to the updates of hubs scores and authorities scores.

7. Future Improvement

As you can see from the above the precision and recall value are not very high. Here are some thoughts that might be helpful for improving the results

7.1 Add weight to the HITS update

In my implementation, I used a edit history as an edge between articles and contributors, and treated the each edit equally. But in fact, in the metadata, there is an attribute of each revision, which is *byte*.

This attribute reflects the bytes of edit a contributor made in this revision. Intuitively, good contributors would write more text to good articles. And good articles are mostly written by good contributors.

In this case, *byte* can be used as weight of the edges in the updates process.

7.2 Consider the lifecycle of edits

In Wikipedia, each edit is justified by the editors later. If the later editors think a specific edit should be removed. Then this might be a low-quality edit.

Some edits may not survive a long time since they are low-quality edits, while good edits may exist very long. If we can detect short-life edits and long-life edits, then we can change the weight of edge of edits in the HITS algorithm.

This may involve in researching a time window of edits, analyzing weather this edit still alive at the end of this time window.

Even further, we can adjust the weight based on the lifetime length within this time window. The longer the edit survives, the higher quality it has.

8. References

[1] "comScore MMX Ranks Top 50 US Web Properties for August 2012". comScore. September 12, 2012. Retrieved February 6, 2013.

[2] Cohen, Noam (February 9, 2014). "Wikipedia vs. the Small Screen". The New York Times.

[3] Wikipedia. Data dumps of Wikipedia. https://meta.wikimedia.org/wiki/Data_dumps

[4] Wikipedia. Size of Wikipedia. https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

[5] Wikipedia. WikiProjects. https://en.wikipedia.org/wiki/WikiProject

[6] Wikipedia. Minor edit. https://en.wikipedia.org/wiki/Help:Minor_edit

[7] Wikipedia. Namespace. https://en.wikipedia.org/wiki/Wikipedia:Namespace

[8] Christopher D. Manning Prabhakar Raghavan Hinrich Schütze. Introduction to Information Retrieval.

[9] Wikipedia. Bots. https://en.wikipedia.org/wiki/Wikipedia:Bots

[10] Wikipedia. List of bots by number of edits. https://en.wikipedia.org/wiki/Wikipedia:List_of_bots_by_number_of_edits

[11] Meiqun Hu, Ee-Peng Lim, Aixin Sun, Hady W. Lauw and Ba-Quy Vuong. Measuring Article Quality in Wikipedia: Models and Evaluation

[12] A. Orlowski. Wikipedia founder admits to serious quality problems, 2005. Published online: 18 October 2005 http://www.theregister.co.uk/2005/10/18/wikipedia_quality_problem.

[13]B. T. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In Proc. of WWW'07, pages 261–270, 2007.

[14] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5):604–632, 1999.