Numeric pattern in URL and its modification.

Roman HUJER* and Atsushi KANAI**

I going to observe a certain set of URLs. In each URL I will try to search for numeric pattern. If I find an occurrence, I'll change the pattern instance and compose back the changed URL. Then I will verify, if the modified URL is a locator of another resource. I will provide statistic in how many cases I found a pattern in URL and in how many cases the modified URL led to another resource.

URL における数字パターンとその派生

Roman HUJER*, 金井　敦**

Web ページを見るためにはその URL を知る必要があり、また、検索エンジンなどでは探し出せない Web ページも多数存在している。そのようなページは存在するが参照できないページであり、無駄になっている。そこで、既知の URL から新たな URL を探し出す手法を提案する。具体的には、URL の数字パターンを検出し、その数字パターンを変化させることにより、新たな、URL を作り出す。本報告では、手法の提案に加え、実際に URL を生成し、どの程度の URL を生成できるか、生成した URL がどの程度有効であるかを実際に検証する。

*Czech Technical University in Prague　** 法政大学

1. Motivation

First It shall be explained what is the purpose of searching for the patterns.

Pagerank algorithm[1] does rank a web page and then distributes the rank to another web pages. The score is redistributed based on hyperlinks found in the ranked page. It is redistributed equally. Each page which is target of the url gets the same part of rank of the source page. So all hyperlinks are considered equally important.

From practical experience everyone knows that not all hyperlinks in a web page are not equally important. Some hyperlinks do serve for navigation, some do serve for miscellaneous function and only few of them are really part of the content. Only few hyperlinks are essential ones, but they got equal portion of rank as the rest.

When you think about previous paragraph you can already find certain classification outlined. Human brain naturally divides the links in the page into classes such as menu or content. The idea is to divide all the hyperlinks in a web page into several classes and then assign the classes with different importance. So the hyperlinks would not get equal portion of pagerank anymore. They will get rank based on their importance. But how to define the classes?

This possibility was already explored by Cun-He, Li; Ke-qiang, Lv in their article “Hyperlink classification: A new approach to improve PageRank”[1]. They looked if a hyperlink points to the same domain and if the hyperlink is part of navigation. This resulted in 4 classes. They assigned each class with different weight and were able to prove that this classification improves the Nutch search engine.

1) Word Pagerank usually implies indexing by google. But there are many more search engines which do work with the same principle.

2. Introduction

I look for different approach of creating the classification. I believe that url itself contains more information than just a locator to a resource. I do look into the syntax of an url and try to extract the information, which do url contains aside the locator. There could be found certain substrings that are repeating in hyperlinks. But not literary. In certain patterns. I want to search for these patterns. Identify them, describe them and quantify their frequency. For this purpose I use program called Sumid.
The patterns have another interesting consequence. In the Internet is a lot of resources, which cannot be accessed from another resource via hyperlink. You can retrieve them only when you know exact url. Also there are resources, which are accessible in standard ways, but they could be also accessible in different ways. And it could be beneficial to access them in another way. Pattern could be helpful in accessing the resource in other way. That means pattern could help us discover previously unknown resources.
The easiest example of a pattern is the numeric pattern.

2.1. Research questions:

How often we could identify a numeric pattern in a url?
How often could change in url according to pattern lead to different resource?

2.2. Hypothesis:

The numeric pattern can be identified in url very often.
In several cases modification of url can lead to a different resource.

3. Name and Purpose of 1M project

Name 1M project is inspired by Operational research. In simplex algorithm is in some cases used so called "M-number". M-number is generally a big number. So 1M project goal is to go trough 1M of hyperlinks[2]. Word "project" implies that 1M project is just a part of "The Analysis of possibility patterns appearance in url"[2].

In motivation I state without any proof that many urls do contain a pattern based just on my observation. 1M project is my way how to get a broader base for my next steps. Of course it is still an observation, but done in larger scale. It is suitable for statistics, but still it is not a proof. So I do not claim to either confirm or deny any ideas proposed in motivation. Primary purpose is to get an insight, how hypothesis of "The Analysis of possibility patterns appearance in url" work for the simplest pattern.

4. Input data

For the analysis input data have to be collected fist, a list of links. Due to need of large amount of hyperlinks an automated way of their collection was employed. Variety of hyperlinks was also desired. So the traditional crawler, looping over a page and collecting all urls that it could find, wouldn't suit the purpose.
After observing some possibilities I decided to use a web-service provided by digg.com. Digg is kind of a social network. It collects resources in web based on user preferences. If a reader considers a web page useful, he diggs it. The more diggs the page receives, the more useful is considered and the more is recommended to other users.
With this approach various enough data will be obtained. These data do already carry some signs of behavior of users in the Internet. Thus the input data are not random.

2) Currently is M set to 2500, but aim is to have much higher M number.

I have to consider the question, if my input data shall be random or not. Different input data will produce different outputs. Different frequency of pattern occurrence will be found. It is impossible to search through the whole Internet. Thus certain subset always have to be selected. I consider better to go for preselected subset[3].

Table 1: Counts of hits frequency

Count of hits	Frequency
0	1103
1	914
>1	471
>2000	73

Internet is ever-changing, but I want to keep my input data stable in order to have comparable results every time the analysis is re-run. This brings an issue that linklist is getting old and many hyperlinks do not work anymore.
The issue is illustrated by the table on the left. From 2500 hyperlinks more than 1100 is already obsolete.

5. Sumid operation

The core of 1M project is conducted with program called Sumid. Sumid takes every url from linklist as a seed and expand it to a tree. First, sumid search trough url, if it contains a pattern. Then each pattern instance is modified, which results in creating a new url. Then Sumid verifies, if this url leads to another resource. Then is changes the url again.
In every url the pattern instance could be found at multiple places and each instance could be modified multiple times. Thus for each seed could be created thousands of new urls to verify. This makes the procedure very time demanding. I have to employ certain technical constraints. First Sumid stops modification of a pattern occurrence after some negative results of verification. But this is not sufficient. So there have to employed limits of count of pattern modification for each pattern occurrence as well as per seed[4].
Verification of resource existence seems like trivial matter, the resource either exists or not. So the result of verify operation for every particular newly created url could be either hit or miss. But in real world servers do response in various ways. The non-triviality of a decision between hit and miss was for me the biggest surprise in the 1M project. The easiest variant of server response is an error response. When server returns a response 4xx or 5xx, it can be easily stated that result is a miss. But not always server returns clear status response and often returns OK even if requested resource does not exist.
With resources that are of an different MIME type than a hypertext, it can be easily uncovered miss with checking MIME type. For hypertext this rule unfortunately does not work, because the incorrect response is always hypertext.
There could be also observed many cases when a url returned resource differs from resource requested. Usually this means that Sumid was redirected from the requested page to some common page, which is doubtlessly a miss. But in some cases that could mean that another full sensible resource was retrieved. When I mark all responses with url

Table 2: Different kinds of misses
This table illustrates how are the misses structured. Please note that no miss was identified from the http status response.

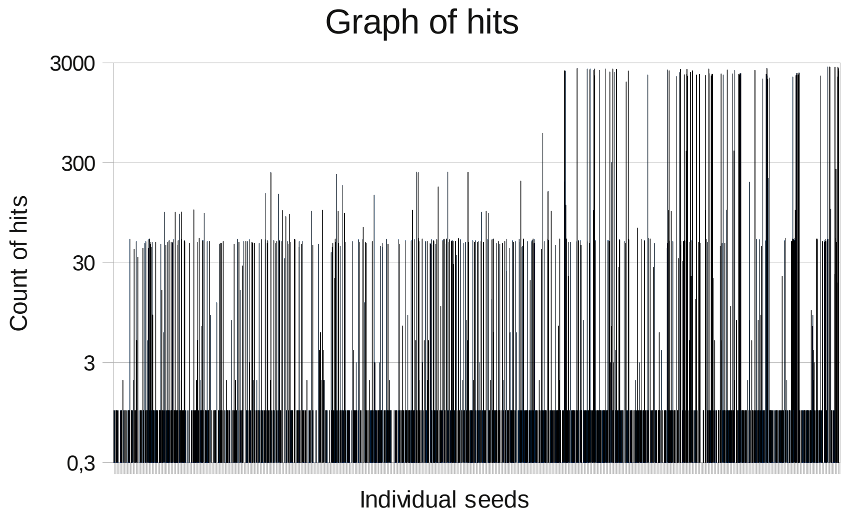
	All Patterns found count	All misses count	Error resp. miss count	Different url miss count	Status resp. miss count	zero Length miss count	Hits
Total count	307793	111237	95596	15272	0	369	198871
Perc. of all patt. found	N/A	36,14%	31,06%	4,96%	0,00%	0,12%	64,61%
Percentage of all misses	N/A	N/A	85,94%	13,73%	0,00%	0,33%	N/A
Count per seed	123,711	44,709	38,423	6,138	0,000	0,148	79,932

3) Preselected by diggs, by users preference.
4) If the limit to url modification by pattern is set to 50, it applies to 2% of all patterns.

different from the requested one as misses I might loose a few hits, but generally I avoid counting misses as hits. So far I have no method how to decide more accurately, when url in response differs from requested one.
Last category of misses I found so far is a server response with zero sized reply. Doubtlessly a miss.
When none of rules mentioned above does not apply to the server response I consider the response to be hit. This is not accurate, because some of the misses could bypass these filters. But it gives me statistically small enough error.

6. Output data

Based on what is for me the result operation I define also overall outputs. Basically I do categorize and quantify server responses for modified urls. I do count how many patterns I found (or actually how many modifications I did). Certainly I count how many times a url modification led to hit. For misses I do count all particular kinds of misses. This helps me produce the statistics.
I also can save to disk the resources which I received as a response from servers in order to check results manually.



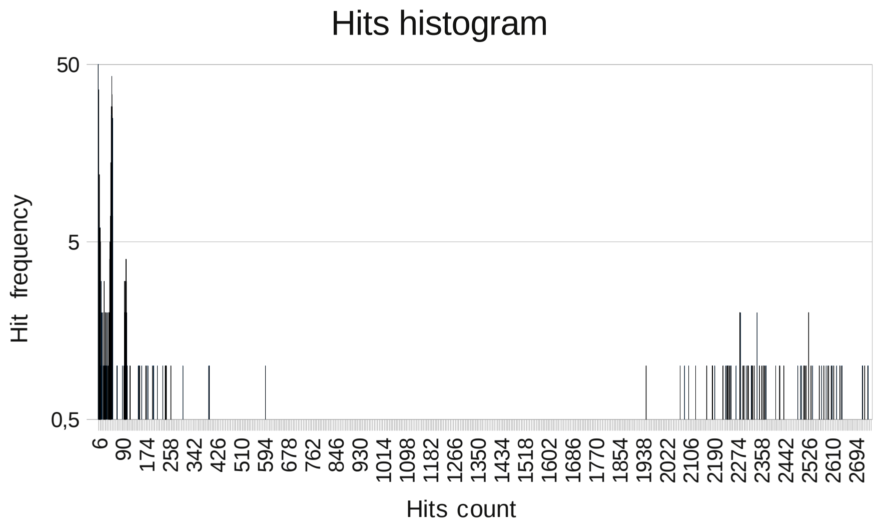
6.1. Outputs and Statistics for M=2500

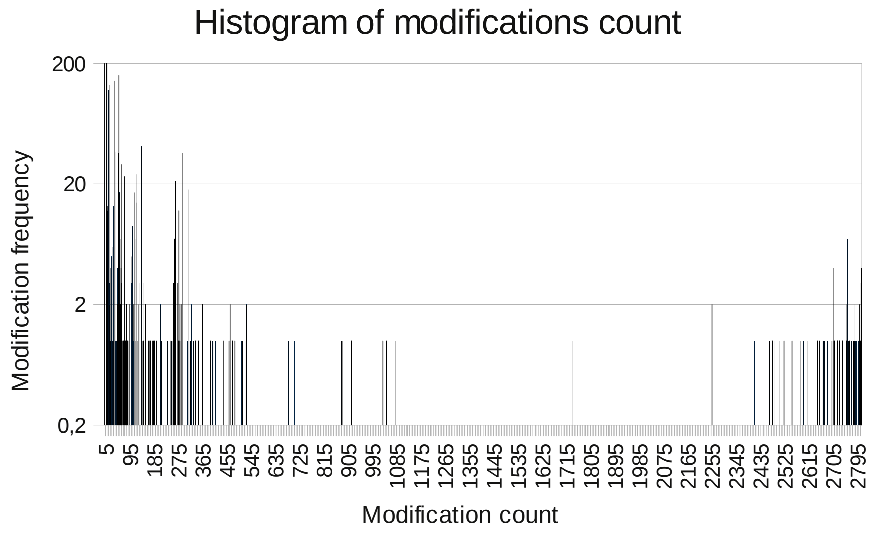
As was already stated Sumid currently operates with quantity of hyperlinks around 2500. The aim is to get the quantity of higher-order number.
The minimal computation time for M=2500 is ca 8 hours. But in different configurations might exceed several days. The main cause of the length of the computation is the request response/time. Taken into account 100 modifications per url (123 shown in table below), we get 250000 request/responses for this quantity. But with the limiting value it would be 2.5 milion.
Sum of just request/response time for M=2500 is about 20 hours. But in multithreaded environment is possible to to whole computation in cca 8 hrs. The sum of content-length is several gigabytes. The maximal speed of computation is 330 seeds per hour, but doesn't grow linearly.
Some of the results of computation with M=2500 were already shown in table2. Following table shows the rest of the results.

Table 3: Verification of modified urls.
This table clearly show that simple average is insufficient to assess the pattern occurrence in url. Median gives better insight. Also is important to filter out the hyperlinks which are already obsolete.

	Pattern Found	Pattern NOT found	Hits	Misses
Total count	307793	407	198871	111237
Average per seed	123,711	0,164	79,932	44,709
Median per seed	8		1	
Average per seed (hit>1)			420,291	
Median per seed (hit>1)			49	

6.2 Histograms of hits and modifications





7. Manual miss check

As I mentioned before, it is not trivial to make a difference between hit and miss. But that applies only to a programmatic way to make the difference. There is also another way. Simply open the seed url and modified url in web browser and I'll see instantly if result is hit or miss. Or in other words - manual miss check. This is basically the way how did I develop the filters for misses. But this way has one limitation. It cannot be used in large scale.
I picked several hyperlinks processed by Sumid and checked manually if the results generated by Sumid are accurate. From this point of view I could divide the hyperlinks in two categories. The hyperlinks, which do contain only one pattern occurrence were assessed accurately by Sumid. The hyperlinks which do contain more than one pattern occurrence showed various results. Some were assessed correctly in all pattern occurrences. Others marked misses as hits in one pattern occurrence and were correct in another. In some cases was Sumid's assessment completely incorrect.

8. Issues and challenges

First issue is that this analysis does focus only on one kind of pattern, numeric. Aside the results I got for numeric pattern I don't have similar data for any other pattern. Further challenge is to find, describe and quantify another patterns.
I also referred about in-triviality of deciding between hit and miss. The challenge here is to observe the results and make the decision more accurate.
Among technical issues is most prominent the performance of the analysis. Every response takes some time, which has tremendous impact, when I sending requests in large scale. This leads to small number M and the statistics is not as good as it could be.
Also I running into issues with keeping the same linklist for long time and also more generally which set of hyperlinks I should include in it.

9. Conclusion

I certainly could say that numeric pattern is important component of an url. With numeric pattern found in 80% of urls and with 20% of urls whose modification leads to discovery of another resource, I consider **research questions answered** and the **hypothesis** of this article **confirmed**.
It also shown some interesting points like decision issues between hit and miss and unexpected count of url seeds with more than 2000 hits. It also clearly points out the further challenges and ways to direct further research on this topic.

10. References

[1] Cun-He, Li; Ke-qiang, Lv
Hyperlink classification: A new approach to improve PageRank
IEEE COMPUTER SOC (2007)

[2] Roman Hujer
The Analysis of possibility patterns appearance in a Uniform resource locator.
CTU in Prague (2009)

11. Used software

Debian GNU/Linux 6.0 squeeze
Python 2.6.6
Eclipse SDK 3.6.0
PyDev 1.6.2
Sumid 0.25
OpenOffice.org 3.2.1