Popular search term analysis on Wikipedia: 2007~2016

Abstract

Wikipedia is the largest and most popular general reference work on the Internet [1][2][3]. The logs of their pagecounts data are publicly available and expand from late 2007 to the end of 2016. This serves as a data source for some interesting exploratory analysis. In this research paper, we examine a set of popular search terms and make a few observations. (placeholder for actual findings)

1 Introduction

Wikipedia, launched in January 2001, became an online substitution of physical encyclopaedia along with the boom of the Internet. It has a unique model of free editing access for all users, which contributed to its fast growth. As of February 2014, Wikipedia has over 5 million articles. Its 18 billion page views and 500 million unique visitors each month are equally remarkable [4].

In this paper, we try to analyse the *pagecounts* data log on Wikipedia from 2007 to 2016. The term *pagecounts* is used in Wikipedia as an approximate measure of hourly page clicks. The two main aspects to investigate are: a) the total *pagecounts* over time, and b) popular search terms over time.

In Section 2 we briefly describe the *pagecounts* data model and the data set used for investigation; in Section 3 we discuss the methodology based on the MapReduce framework; in Section 4 we present the analytical findings as well as visualizations of the results; in Section 5 we explore some flaws in the study and consider related following work; we conclude in Section 6.

2 Dataset

As discussed before, the data used in this investigation is called *pagecounts*. This roughly translates to a page access, done by either webpage interaction or search from human users, and alternatively by HTTP calls made in code. In 2.1 the data format of *pagecounts* is presented and discussed, and in 2.2 we explore the archive layout of the Wikipedia dataset. These are crucial characteristics of the data structure and are essential in the process of obtaining and understanding the data sample of our study.

2.1 Data format

Officially, the data format of pagecounts is as following [5]:

domain_code page_title count_views total_response_size

Domain_code: domain name of the request, abbreviated. This is not used in our investigation.

Page_title: usually the normalized part after "/wiki/" in the request URL.

Count_views: number of times this page has been viewed in the respective hour.

Total_response_size: the size of response sent back. This is not used in our investigation.

The page title field in particular is our key focus of the investigation, since it contains the keyword searched. However, there are a few reasons that prevent us from using it directly. First of all, Wikipedia has been expanded into many languages and a wide range of these languages do not use the English alphabet. Therefore a page title may not be directly included in the URL, as there is a non-trivial part of the data that contains Unicode characters. In our investigation, we simply ignore any occurrence of non-alphanumeric characters present. We are aware that this is a point for future improvement, and mention it once again in Section 6. Secondly, the page title is simply set as the later segment of the requested URL. The issue is that the requested URL could contain project name, followed by colon, and then by many '/' separated words. These expressions might be irrelevant, and it is our responsibility to extract the desirable word and pass it down for analysis. This is addressed in Section 3.1.1.

2.2 Archive storage layout

All *pagecounts* data used in this investigation is downloaded from Wikipedia data dump storage [5]. Naturally, the data is partitioned. Because of the large access amount on Wikipedia, it is compressed and partitioned into the scale of hour. To specify a partition, one would need year, month, day, and hour of that day. An example would be "pagecounts-20160101-080000.gz". More specifically, this maps to the *pagecounts* data in January 1st, 2016, during the hour from 7 to 8 in the morning.

Some of the properties of this data storage layout pose difficulties to our investigation. First, the sheer size of data is overwhelming for the cluster on which we run the investigation. This is inevitable since data has to be unzipped and put into HDFS to be analysed. By a rough account, one day of uncompressed data has a size between 7 to 10 gigabytes. How this is handled will be covered in 3.2.2. Additionally, it is not safe to assume that the last four digits, which represent the minute and second in the time stamp, are all zeros. We note that this is true in most of the cases, but we do observe a fair amount of data that has a non-zero second part. An example would be "pagecounts-20140101-110011.gz", which means that the file is not generated at the perfect hour but 11 seconds afterwards. This makes downloading harder since there is no guarantee on the filenames. We will further discuss this finding and corresponding workarounds in Section 3.2.2.

3 Implementation

In principle, we carry out the analysis using MapReduce framework. We pair the methodology with some custom scripting to schedule jobs and further process the results to generate tables and graphs. The last part is discussed in 4.1. In this section, we focused on the MapReduce job and the scripting that enables it to analyse the data. Due to various considerations, we use the first day of January and June as representatives for each year. The exception is the year 2007, which only has data available for the month of December. This makes our investigation steps faster, simpler, and still capable of generating meaningful results.

3.1 MapReduce

MapReduce is chosen as the data analysis technology over Spark for our investigation for a couple of reasons. First, it is highly scriptable, which makes it fit better into our scripting-predominate approach. Second, as we mentioned previously, the data set has to be processed in smaller batches due to its scale and the capacity of the cluster. In light of this limitation, Spark's advantage of keeping data structures in memory for faster computation is undercut. Last but not least, Spark's interactive nature is unnecessary for us since we scripted the instigation steps.

In Section 3.1 we discuss the detail of Mapper, and in Section 3.2 the reducer. The master control class is trivial in this investigation, but for completeness the number of reducer is set to 10 for all jobs. Again we will discuss why this is trivial.

3.1.1 Mapper

The mapper used in the investigation extracts searched terms as keys, and number of views in each hour as values in the mapping phase. In Section 3.1.1 we discuss some sanity checking mechanism applied, and in Section 3.1.2 the exact key value pair produced is presented.

3.1.1.1 Sanity Checking

The sanity checks are primarily for the page_title field, which has great variation and many different formats. The following are the checks we deployed to best extract the meaningful search term. Note that these mechanisms are interleaved in implementation, to account for all possible formats of page_title. Implementation-wise, a function *getTerm()* is constructed, which takes the plain page_title and returns a meaningful search term, or null if it fails the checks.

First, a check disallows any Unicode search terms to be returned. As mentioned before, the target of this investigation is English search terms. If there is a Unicode character is encountered, which is indicated by the presence of "%", a null value is returned.

Second, a check disallows any search term of length 2 to be returned. This is due to a large presence of 2-letter country code in the URL, which is irrelevant to our investigation. Same as before, a length 2 search term will be discarded and a null value is returned.

Lastly, because the page_title cab comprises of a project name and a path to the file under such project, there should be a check for that as well. For a project/path term, indicated by the presence of ':', only the project name is returned. The reasoning is that we do not want different paths to disperse a would-be significant project. Otherwise, we just return the word before the first '/' of the path.

3.1.1.2 Mapper output

Using the *getTerm()* function, a meaningful search term is extracted, which will be the key of mapper output. For the value, the *count_views total* field is directly used after casting into an Integer. No hourly page view for a term is over the Integer limit in Java, so using Integer as key is sufficient. These key/value pairs are passed down for further analysis by the reducers.

3.1.2 Reducer

The job of the reducer is fairly straightforward, namely to accumulate the view counts for each search term and output them when done. In other words, what is left after the reduce step is a collection of key/value pairs. The key is the search term, and the value is the total view count for each term. Note that the output is unsorted in the number of view counts, and an order needs to be established by further operations.

3.2 Scripting

MapReduce, though powerful in analysing, needs to be properly scheduled with potential changing input. In this Section we discuss the three main scripts, all in bash, used to facilitate the analysis. The script *run_mapred.sh* is a wrapper to run MapReduce for debugging, *day_count.sh* is used for generating analysis for a year, and *year_count.sh* is for yearly analysis.

3.2.1 run_mapred.sh

Although MapReduce has an architecture that is easy to write and understand, there are a few operations involved in actually running MapReduce jobs. This script is designed to serve the following purposes. First, it removes the output directory in HDFS as requested by MapReduce, compiles the java MapReduce code into a jar, runs it as a MapReduce job, and generates the output as a text file into a user file system and sorts it by page counts. This script is used mostly for debugging reasons, and in the actual analysis we used part of this script but not its entirety.

3.2.2 day_count.sh

As the name suggested, this script is responsible for generating a daily page count report. It takes the year, month, and day as arguments. Before anything, it destroys the input directory in HDFS and re-creates it immediately to make space for the new data. As mentioned before, due to the large size of data, we can only analyse *pagecounts* at the granularity of days.

After the new clean directory is created, it starts to download compressed data from the archive using *wget*. There should be 24 zip files to download, one for each hour. To account for the unstable seconds value in the filename, this script loops from '00' to '13', which is the observed largest seconds, and try to download all of them. Naturally most of them will fail, but that does not affect our analysis. The most important point is that we do not miss an hour's data. Upon getting a zip file, the script un-compresses it and puts it into HDFS.

After all the inputs are put in, a new MapReduce job is created to analyse them. This part is very similar to *run_mapred.sh*, except that the final output text file is time stamped given the input arguments, so many output files do not override each other.

3.2.3 year_count.sh

This is the ultimate script run for the entire investigation. It basically just calls *day_count.sh* repeatedly, with all days of interest. For the year 2007, we only analyse the day December 10th, and all following years, we analyse January 1st and June 1st for each. After the script is done, 19 text files should be generated, each time-stamped in filename to indicate the day of analysis

4 Finding and Analysis

In this Section we present findings gathered from previous investigations. In Section 4.1 the overall trend over time is discussed, and in Section 4.2 we look at specific search terms and the change of their view counts over time.

4.1 Aggregated *pagecounts*

4.1.1 Search term agnostic pagecounts

Figure 1 depicts the change of overall pagecounts in the past decade. We observe a steady increase from 2007 to mid 2014, a subsequent significant drop to the end of 2015, and a potential rebound since 2016. The increase is explainable since more and more people have access to the Internet globally. The drop in view counts can be a reflection of the Wikipedia donation scandal [6]. However, over some time it seems like view counts are picking back up.

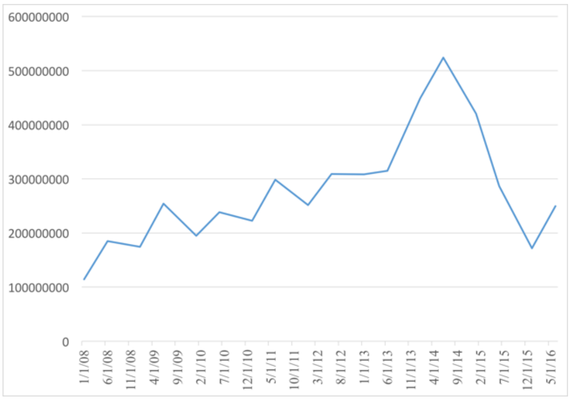

Figure 1 Overall pagecounts over time

4.1.2 Search term specific pagecounts

The key point of interest for this investigation is to see if certain search terms became increasingly popular over time or otherwise. Figure 2 and 3 show the most popular search terms, measured by overall count. The terms “meta” and “Main_page” dominate all the following terms, because they are project names and should accumulate view counts from terms of all paths under them. If we ignore the top 2 terms and look at Figure 3, the distribution is much smoother. What is particularly interesting among many of the top search terms, such as “Pagina_principale”, “index.html”, “Hauptseite”, “Accueil”, is that they all map to the home page of Wikipedia in different languages. This may give some hint in terms of user behaviour. A plausible explanation is that large amounts of users

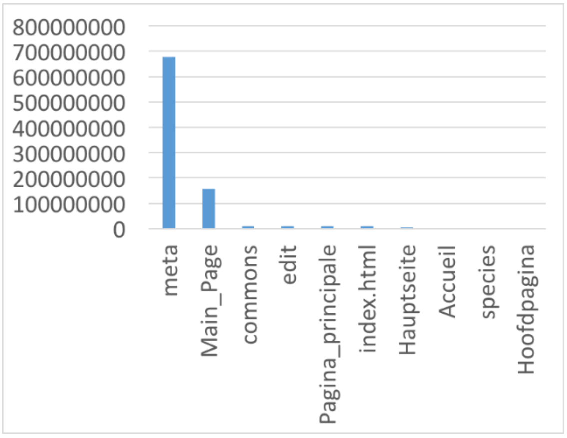

Figure 2 Top 10 overall popular search terms

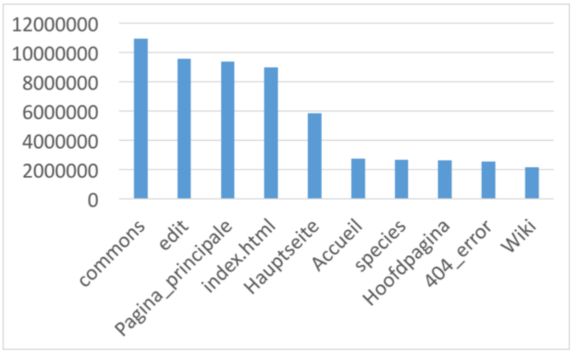

Figure 3 Rank 3 to 12 popular search terms

choose to land in the homepage of Wikipedia, and then proceed to search for specific terms.

4.2 search term findings

In this section we present search term popularity change over time in three contexts: general, company, people. Note that because the top 2 terms have figures so large that makes visualization less clear, we ignore them for this section.

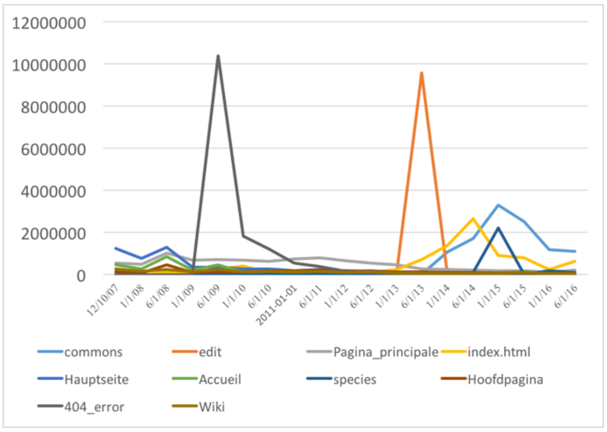

Figure 5 General Search terms popularity change

4.2.1 General search terms

Figure 4 shows the top 10 search terms and their view counts change over time. We observe two huge spikes for term “404_error” and “edit” respectively, although the specific cause is not clear. However, it seems like since 2014, there has been rise in views for a few terms. The terms that experienced a rise in views are “edit”, “commons”, “species”, and “index.html”, three of which are project names. This could suggest an increase in popularity of terms in these three projects.

4.2.2 Company search terms

Shown in figure 5 are popular company search terms and the changes of their view counts over time. Overall, the view counts for these companies display a slight decrease over the decade. Notably however, there are two spikes observed in early 2009

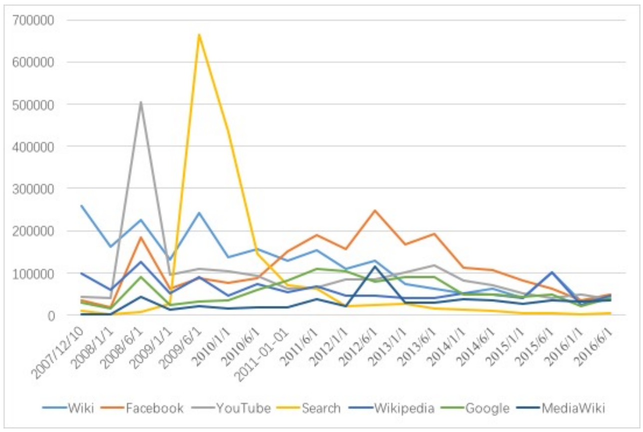

Figure 4 Company Search terms popularity change

and mid-2012. The spike in 2009 is significant but hard to explain. The spike in 2012, on the other hand, is less great in magnitude but more reflective to tech events happening during the period. For example, the sudden rise in views for “Facebook” in early 2012 could come from its acquisition of Instagram in April and its IPO in May [7].

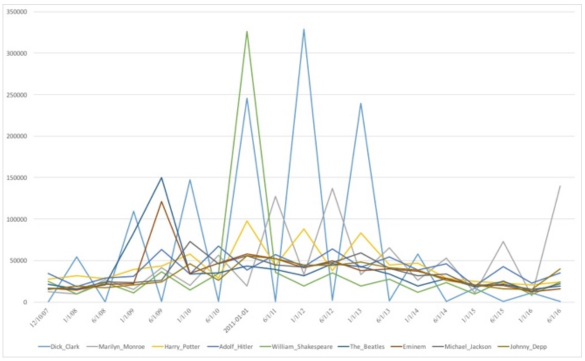

Figure 6 Name-related search term popularity change

4.2.3 People search terms

Figure 6 shows the popularity change of search terms that consist of people's names. Overall, there is a peak during late 2011 and early 2012. Moreover, a general wave pattern is visible in the plot of the view counts. For every name term, we typically see a cycle of high and low counts over every 6 months. The peaks of the waves appear to be term-specific, and we have a few intriguing discoveries in that regard. For instance, the term "Harry Potter" has a surge in its wave at the beginning of the years, whereas the term "Michael Jackson" seems to have its peak located at the middle of the years.

After reasoning the three contexts of terms, the bottom line is unlike what we postured, there does not seem to be a significant popularity change for search terms at least in examined contexts.

5 Future Improvements

During analysis coding and generating reports and graphs, there are a few things that should have been designed differently. In this section, we discuss some shortcomings of the investigation and potential future improvements.

5.1 Extracting search term

In the Mapper, many entries are thrown away. The reason could be either Unicode, or the fact that all paths of a project name are treated as one, etc. In terms of improvements, with more investigations on the *page_title* field we can better extract more meaningful and representative search terms. We would expect to be able to improve the accuracy of our analysis as a result.

5.2 Sampling

In our investigation, we only used 2 days of data as a representative for the entire year. This is mainly due to time constraints. The time used for this investigation, which covers 19 days of data, took roughly 12 hours to finish. This is partly because the data has to be downloaded and swapped out often due to space constraints. In the future, more days should be chosen to represent a year, and there should be some normalization mechanisms in place as well.

5.3 Approach

One of the steps that was especially time consuming when running MapReduce jobs is getting inputs. A reason for this is that the script is trying to download many files that do not exist, and getting meaningless 404 errors. This is necessary because we do not want to missing an hour of data and end up with an incomplete input set. Also, the cluster on which we run MapReduce cannot take too many inputs. For safety concerns, the script is only putting in one day of data at a time and deleting it after processing. This also makes repetition and replication of the analysis slower because the data has to be downloaded and put into HDFS again. In the future, it would be better suited if these inputs and jobs can be done on a cloud platform with more space available and higher computation power.

6 Conclusion

In conclusion, we carried out a rough analysis on Wikipedia's *pagecounts*. We found that the overall trend of views has been increasing until 2014, dropping for the year to follow, and has been on the rise once more since 2016. Similarly, we observe some potential of rise in view counts for a few popular terms, but the majority of popular terms stay flat recently and do not promise a potential growth. Terms under 3 contexts do not show a growth potential either. In all, as of 2016 it seems like Wikipedia has reached a plateau in view counts. However, due to the coarse nature of the analysis, we hope to apply a deeper level of fine-tuning and reasoning in the future.

7 References

1. *Bill Tancer (May 1, 2007). "Look Who's Using Wikipedia". Time. Retrieved December 1, 2007. The sheer volume of content [...] is partly responsible for the site's dominance as an online reference. When compared to the top 3,200 educational reference sites in the US, Wikipedia is No. 1, capturing 24.3% of all visits to the category*. Cf. Bill Tancer (Global Manager, Hitwise), "Wikipedia, Search and School Homework" Archived March 25, 2012, at the Wayback Machine., *Hitwise*, March 1, 2007.
2. *Alex Woodson (July 8, 2007). "Wikipedia remains go-to site for online news". Reuters. Retrieved December 16, 2007. Online encyclopedia Wikipedia has added about 20 million unique monthly visitors in the past year, making it the top online news and information destination, according to Nielsen//NetRatings.*
3. *"comScore MMX Ranks Top 50 US Web Properties for August 2012"*.

comScore. September 12, 2012. Retrieved April 23, 2017.

4. Cohen, Noam (February 9, 2014). "Wikipedia vs. the Small Screen". The New York Times.
5. Page view statistics for Wikimedia projects. Retrieved April 23, 2017.
6. Caitlin Dewey (December 2, 2015) "Wikipedia has a ton of money. So why is it begging you to donate yours?" Washington Posts
7. Timeline of Facebook. Wikipedia. April 10, 2017. Retrieved April 23, 2017.