PROJECT 1: DATA QUERIES OF WIKIPEDIA

Kevin M Conlin

OBJECTIVES

- Find the English Wikipedia article that got the most traffic on October 20, 2020.
- Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article.
- Find what series of Wikipedia articles, starting with Hotel California, keeps the largest fraction of its readers clicking on internal links.
- Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia.
- Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed.
- Run an analysis you find interesting on the Wikipedia datasets we're using.

Find the English Wikipedia article that got the most traffic on October 20, 2020.

Find the English Wikipedia article that got the most traffic on October 20, 2020.

I first downloaded and concatenated all of the hourly page view data for October 20th. However, this gave me 24 records of each page, each with its hourly page views. This file also included all data from all Wikipedia projects (not just en and en.m). To fix this, I created a MapReduce job that took this massive input file and output a file that that included only the English articles, and output them as a daily (article, page views pair).

```
conlink@DESKTOP-93ULVNL:~$ hdfs dfs -ls /user/conlink/pageviews
Found 2 items
drwxr-xr-x   - conlink supergroup          0 2020-10-28 12:00 /user/conlink/pageviews/10-20
drwxr-xr-x   - conlink supergroup          0 2020-10-28 12:40 /user/conlink/pageviews/dailyPageViews_10_20
conlink@DESKTOP-93ULVNL:~$ hdfs dfs -ls /user/conlink/pageviews/10-20
Found 1 items
-rw-r--r--   1 conlink supergroup 4615630043 2020-10-28 12:00 /user/conlink/pageviews/10-20/pageviews-10-20
conlink@DESKTOP-93ULVNL:~$ hdfs dfs -ls /user/conlink/pageviews/dailyPageViews_10_20
Found 2 items
-rw-r--r--   1 conlink supergroup          0 2020-10-28 12:40 /user/conlink/pageviews/dailyPageViews_10_20/_SUCCESS
-rw-r--r--   1 conlink supergroup  140340823 2020-10-28 12:40 /user/conlink/pageviews/dailyPageViews_10_20/part-r-00000
conlink@DESKTOP-93ULVNL:~$
```

Find the English Wikipedia article that got the most traffic on October 20, 2020 (Cont).

After loading the MR output file into a hive database, I retrieved the top ten view pages of 10/20/2020 with the following HQL command:

```
SELECT * FROM 10_20_pageviews
ORDER BY views DESC
LIMIT 10;
```

10_20_pageviews.article	10_20_pageviews.views
Main_Page	5961008
Special:Search	1476831
-	544714
Jeffrey_Toobin	321459
C._Rajagopalachari	210558
The_Haunting_of_Bly_Manor	185139
Robert_Redford	178779
Jeff_Bridges	159163
Bible	151484
Chicago_Seven	149966

Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article.

Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article.

First, I ran a MapReduce job to output a key value pair of (article name, number of occurrences) for every article in which people followed an internal link.

Ubuntu

```
conlink@DESKTOP-93ULVNL:~$ hadoop jar /mnt/c/Users/kevin/OneDrive/Desktop/Git/pj1/WikiInternalLinkMR/LinkCount/target/sc
ala-2.13/WikiInternalLinkMR-assembly-0.1.jar clickstream internal-link-output
2020-10-26 11:52:16,889 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
```

This drastically reduced the size of the data file I would need to query to find a result, i.e. which article had the most users follow an internal link.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	conlink	supergroup	1.32 GB	Oct 26 10:29	1	128 MB	clickstream-enwiki-2020-09.tsv

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	conlink	supergroup	0 B	Oct 27 07:51	1	128 MB	_SUCCESS
-rw-r--r--	conlink	supergroup	46.52 MB	Oct 27 07:51	1	128 MB	part-r-00000

Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article (Cont.)

I then repeated this process on the total pageviews using the page view data from the same time period (September 2020). This was a very laborious MapReduce job, as the total pageview files amounted to roughly 40 GB

```
Map input records=4325550946
Map output records=1486364229
Map output bytes=36619787376
Map output materialized bytes=39592684955
Input split bytes=104400
Combine input records=0
Combine output records=0
Reduce input groups=16674407
Reduce shuffle bytes=39592684955
Reduce input records=1486364229
Reduce output records=16674407
Spilled Records=5306002131
Shuffled Maps =720
Failed Shuffles=0
Merged Map outputs=720
GC time elapsed (ms)=307473
CPU time spent (ms)=25103170
Physical memory (bytes) snapshot=370164768768
Virtual memory (bytes) snapshot=1848327323648
Total committed heap usage (bytes)=344082350080
Peak Map Physical memory (bytes)=529518592
Peak Map Virtual memory (bytes)=2572742656
Peak Reduce Physical memory (bytes)=986562560
Peak Reduce Virtual memory (bytes)=2589093888
```

Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article (Cont.)

After creating tables in Hive for each of the MapReduce outputs, I used the following HQL command to join the two tables on article name, as well as to display the clickthrough rate for the top ten pages in September 2020:

```
SELECT 09_pageviews.aritcle AS Article,
09_pageviews.views AS Total_Views,
link_count.links AS Links_Followed,
ROUND(link_count.links/09_pageviews.views*100, 2) AS Clickthrough_Percentage
FROM 09_pageviews
INNER JOIN link_count ON 09_pageviews.aritcle=link_count.article
WHERE 09_pageviews.views > 1000000
ORDER BY Clickthrough_Percentage DESC
LIMIT 10;
```

Find the Wikipedia article that has the largest fraction of its readers following an internal link to another Wikipedia article (Cont.)

article	total_views	links_followed	clickthrough_percentage
Dune_(2020_film)	1278838	1201459	93.95
Cobra_Kai	2459988	2241751	91.13
COVID-19_pandemic_by_country_and_territory	1207880	1093321	90.52
Schitt's_Creek	1493588	1339942	89.71
Elizabeth_II	1065045	922145	86.58
Sarah_Paulson	1252257	987550	78.86
Supreme_Court_of_the_United_States	1278921	1002716	78.4
2016_United_States_presidential_election	1052232	768124	73.0
Enola_Holmes_(film)	1980000	1356311	68.5
2020_United_States_presidential_election	1150820	749205	65.1

Find what series of Wikipedia articles, starting with Hotel California, keeps the largest fraction of its readers clicking on internal links.

Find what series of Wikipedia articles, starting with Hotel California, keeps the largest fraction of its readers clicking on internal links.

In order to discover which series of articles kept users following links originating from Hotel California, I created a Hive table that contained all of the clickstream data for September of 2020. I then ran the following HQL command:

```
SELECT * FROM clickstream
WHERE previous_article='Hotel_California'
ORDER BY clicks DESC
LIMIT 10;
```

clickstream.previous_article	clickstream.current_article	clickstream.link_type	clickstream.clicks
Hotel_California	Hotel_California_(Eagles_album)	link	2222
Hotel_California	Don_Henley	link	1537
Hotel_California	Don_Felder	link	1519
Hotel_California	Eagles_(band)	link	1335

Based On this output, it was clear that the article that the majority of users linked to from Hotel_California was Hotel_Califonia_(Eagles_Album) I then repeated this process replacing previous article with the current article to follow the series of top articles.

clickstream.previous_article	clickstream.current_article	clickstream.link_type	clickstream.clicks
Hotel_California_(Eagles_album)	The_Long_Run_(album)	link	2127
Hotel_California_(Eagles_album)	Hotel_California	link	2010
Hotel_California_(Eagles_album)	Their_Greatest_Hits_(1971-1975)	link	897
Hotel_California_(Eagles_album)	Eagles_(band)	link	801

Find what series of Wikipedia articles, starting with Hotel California, keeps the largest fraction of its readers clicking on internal links (Cont).

clickstream.previous_article	clickstream.current_article	clickstream.link_type	clickstream.clicks
The_Long_Run_(album)	Eagles_Live	link	1322
The_Long_Run_(album)	Hotel_California_(Eagles_album)	link	654
The_Long_Run_(album)	I_Can't_Tell_You_Why	link	470
The_Long_Run_(album)	Heartache_Tonight	link	327

clickstream.previous_article	clickstream.current_article	clickstream.link_type	clickstream.clicks
Eagles_Live	Eagles_Greatest_Hits,_Vol._2	link	1136
Eagles_Live	The_Long_Run_(album)	link	223
Eagles_Live	Seven_Bridges_Road	link	127

clickstream.previous_article	clickstream.current_article	clickstream.link_type	clickstream.clicks
Eagles_Greatest_Hits,_Vol._2	The_Very_Best_of_the_Eagles	link	996
Eagles_Greatest_Hits,_Vol._2	Eagles_Live	link	186

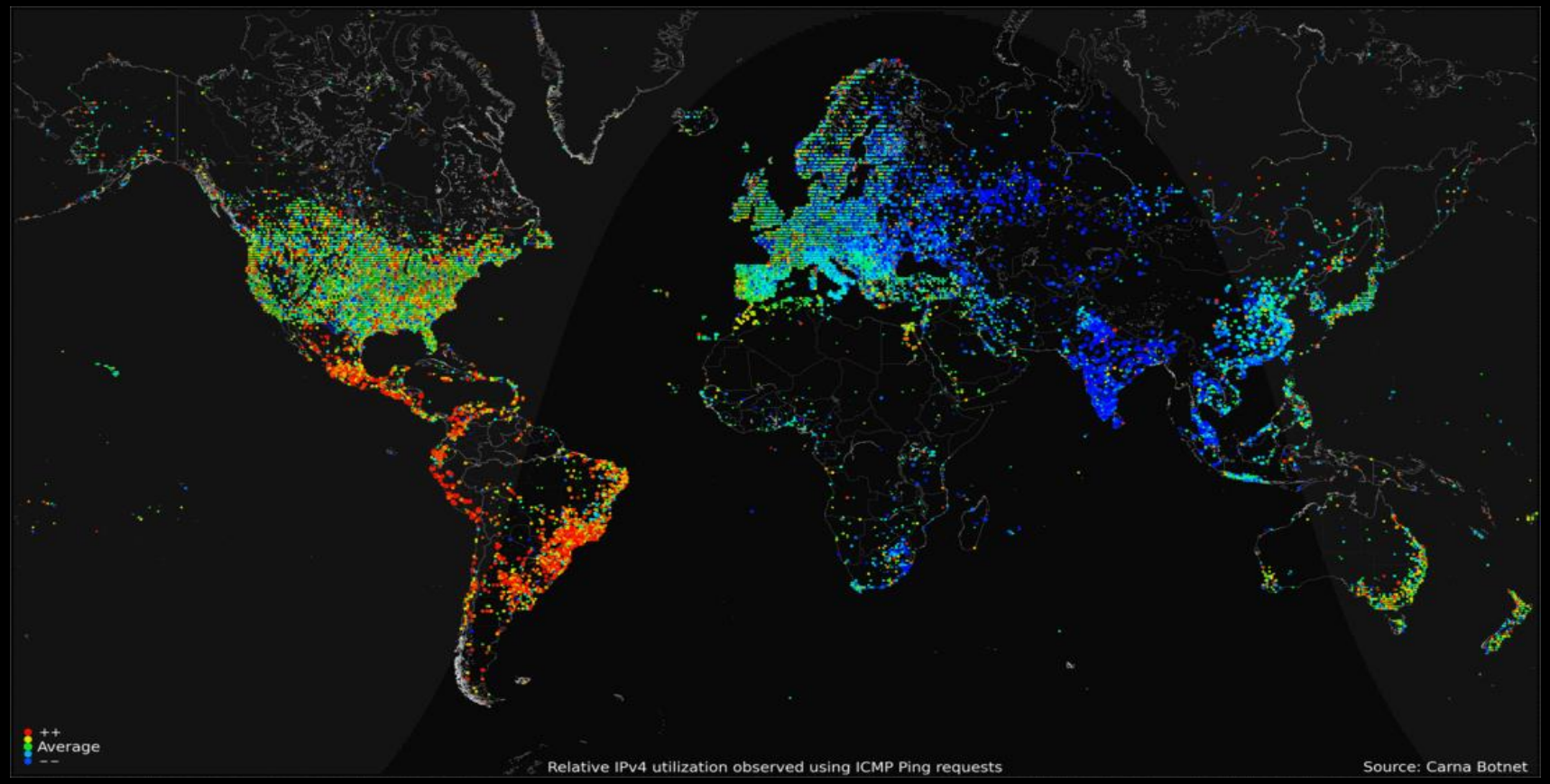
Based on these 5 HQL queries, I found the most common series of articles to be followed from Hotel California to be: **Hotel California => Hotel California (Eagles album) => The Long Run (album) => Eagles Live => Eagles Greatest Hits, Vol. 2 => The_Very_Best_of_the_Eagles**

It is important to note that this method does not account for the possibility that people could arrive at these intermediate articles through other means, but instead assumes that people arrived at each new queried article via link from the previous article's query.

Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia.

Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia.

In order to find articles with more popularity in each of the three countries, I focused on isolating the hours in which each of these countries is most likely to be using the internet. Using data provided by Carna Botnet, I determined internet traffic in each country corresponded most heavily with daylight hours.



Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia (Cont).

Using my MapReduce for page views, I isolated just the relevant information for my search (pageview traffic on en and en.m articles), loaded each countries individual activity into a corresponding table on Hive, and ran an HQL query across the three tables:

```
SELECT us_activity.article AS article_name,
us_activity.views AS united_states_views,
uk_activity.views AS united_kingdom_views,
au_activity.views AS australia_views
FROM us_activity
INNER JOIN uk_activity ON uk_activity.article=us_activity.article
INNER JOIN au_activity ON au_activity.article=us_activity.article
ORDER BY us_activity.views DESC
LIMIT 25;
```

Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia (Cont).

article_name	united_states_views	united_kingdom_views	australia_views
Main_Page	3502059	3483323	2838781
Special:Search	870115	900024	679682
-	316346	331540	250520
C._Rajagopalachari	211820	209067	402
Jeffrey_Toobin	170950	147620	192879
The_Haunting_of_Bly_Manor	109227	80802	110539
Sokushinbutsu	104346	1260	459
Chicago_Seven	83446	60463	95757
Bible	81275	82847	75149
Three_Red_Banners	77797	15065	151
Robert_Redford	77723	91547	96999
Deaths_in_2020	73092	64318	52903
The_Trial_of_the_Chicago_7	67382	52702	68698
Peter_Madsen	67085	47444	656
Harshad_Mehta	65657	80640	46253
Jeff_Bridges	64358	78810	84728
Andy_Burnham	63288	52189	10904
QAnon	62960	54879	53123
2016_United_States_presidential_election	61147	44415	47720
Kamala_Harris	57950	40521	30137
Joe_Biden	57168	44732	50449
Abbie_Hoffman	57053	38762	73035
Hunter_Biden	56698	41744	52377
Murder_of_Kim_Wall	55705	45314	614
Rush_Limbaugh	55666	44056	31245

Find an example of an English Wikipedia article that is relatively more popular in the UK. Find the same for the US and Australia (Cont).

Based on the results from the HQL query shown on the previous slide, I determined articles related to US Politics are relatively more popular in the United States, Articles relating to Hollywood/Entertainment are more popular among Australian Wikipedia users, and UK Wikipedia users are the most likely to use the internal search features of Wikipedia.

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed.

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed.

```
SELECT
FORMAT_NUMBER(AVG(revision_seconds_to_identity_revert/3600), 0) AS HOURS,
FORMAT_NUMBER(AVG((revision_seconds_to_identity_revert%3600)/60), 0) AS MINUTES,
FORMAT_NUMBER(AVG((revision_seconds_to_identity_revert%3600)%60), 0) AS SECONDS
FROM wiki_edits
WHERE revision_seconds_to_identity_revert > 0;
```

hours	minutes	seconds
23	18	28

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed (Cont).

```
SELECT
FORMAT_NUMBER(AVG(views)/30, 0) AS average_views_per_day,
COUNT(aritcle) AS  total_pages
FROM 09_pageviews;
```

average_views_per_day	total_pages
14	16674407

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed (Cont).

```
SELECT
(AVG(revision_seconds_to_identity_revert)/86400) AS average_days_to_revert
FROM wiki_edits
WHERE revision_seconds_to_identity_revert > 0;
```

```
+-----------------------------+
|  average_days_to_revert     |
+-----------------------------+
|  0.94546023351462 09        |
+-----------------------------+
```

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed (Cont).

```
SELECT
FORMAT_NUMBER(14*(AVG(revision_seconds_to_identity_revert)/86400), 2)
AS average_page_views_before_revision
FROM wiki_edits
WHERE revision_seconds_to_identity_revert > 0;
```

average_page_views_before_revision
13.24

Analyze how many users will see the average vandalized Wikipedia page before the offending edit is reversed (Cont).

NOTE: One important caveat to note in the previous analysis is that the revision_seconds_to_identity_revert field, which provides the amount of time that passes between an initial revision and the time in which that revision is reverted to a previous state, has some major inconsistencies in its original format. Though the documentation implies that this field is a number of seconds, many of the values in the original dataset are very large negative numbers. As a form of simplification, I omitted al fields of this data that were not positive values, however it is difficult to know what the values of those negative fields were intended to be, and thus my final result may be skewed by the omission of this data.

Run an analysis you find interesting on the Wikipedia datasets we're using.

Analyze the percentage of mobile views across various pages to look for any correlation between consumption medium and article content.

For this objective, I analyzed the percentage of mobile views for various pages to attempt to find a relationship between content type and viewing methods.

After altering my MapReduce for page views to output only mobile page views, I created a table for mobile page views and ran two HQL queries to return the highest and lowest percentages of mobile views.

Analyze the percentage of mobile views across various pages to look for any correlation between consumption medium and article content (Cont).

```
SELECT 09_pageviews.aritcle AS Article,
09_pageviews.views AS Total_Views,
ROUND(mobile_views.views/09_pageviews.views*100, 2) AS Mobile_Percentage
FROM 09_pageviews
INNER JOIN mobile_views ON mobile_views.article=09_pageviews.aritcle
WHERE 09_pageviews.views > 1000000
ORDER BY Mobile_Percentage DESC
LIMIT 10;
```

article	total_views	mobile_percentage
Robert_F._Kennedy_Jr	1112876	99.88
XXXX	2056847	97.9
Dennis_Nilsen	3564441	87.73
S._P._Balasubrahmanyam	2387782	86.04
William_Zabka	1346183	84.37
Ralph_Macchio	1260530	83.88
Sarah_Paulson	1252257	83.15
Nurse_Ratched	1266740	82.7
Dan_Levy_(Canadian_actor)	1191431	80.88
Tyler_Herro	1158733	80.51

Analyze the percentage of mobile views across various pages to look for any correlation between consumption medium and article content (Cont).

```
SELECT 09_pageviews.aritcle AS Article,
09_pageviews.views AS Total_Views,
ROUND(mobile_views.views/09_pageviews.views*100, 2) AS Mobile_Percentage
FROM 09_pageviews
INNER JOIN mobile_views ON mobile_views.article=09_pageviews.aritcle
WHERE 09_pageviews.views > 1000000
ORDER BY Mobile_Percentage ASC
LIMIT 10;
```

article	total_views	mobile_percentage
Bible	3170711	2.35
F5_Networks	1487955	8.4
Portal:Current_events	1234802	25.16
Special:Search	41915305	39.58
YouTube	1289577	41.7
COVID-19_pandemic_by_country_and_territory	1207880	42.88
Periodic_table	1376694	42.92
Deaths_in_2020	3316200	47.18
COVID-19_pandemic	1376339	49.75
Mulan_(2020_film)	3239724	51.19

Analyze the percentage of mobile views across various pages to look for any correlation between consumption medium and article content (Cont).

Based upon the results of the two HQL queries I ran, I believe a loose correlation can be drawn between the nature of an article's content and the medium through which people are most likely to use when viewing it. Based on the high prevalence of more light-hearted/entertainment-based articles in the first query, it appears these types of topics lend themselves more readily to mobile viewing.

In contrast, those article that had a higher percentage of desktop users seemed to correlate with more serious topics, such as statistics relating to the coronavirus or articles related to work/study topics like the periodic table of elements.

SOURCES

- https://dumps.wikimedia.org/other/analytics/
- https://dumps.wikimedia.org/other/pageviews/readme.html
- https://dumps.wikimedia.org/other/mediawiki_history/readme.html
- https://dumps.wikimedia.org/other/clickstream/readme.html
- https://darknetdiaries.com/imgs/carna.gif

GITHUB REPO

https://github.com/Kevin-Conlin/pj1

Questions?