# OpenGeist: Insight in the Stream of Page Views on Wikipedia

Maria-Hendrike Peetz
ISLA, University of Amsterdam
M.H.Peetz@uva.nl

Edgar Meij
ISLA, University of Amsterdam
edgar.meij@uva.nl

Maarten de Rijke
ISLA, University of Amsterdam
derijke@uva.nl

## ABSTRACT

We present a RESTful interface that captures insights into the zeitgeist of Wikipedia users. The system is an interface for clustering and comparing concepts based on the time series of the number of views of their Wikipedia page. The functionality is motivated by three use cases, ranging from technical novices to expert users and we also provide two real-life example applications.

## Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.4 Systems and Software;H.3.5 Online Information Services

## General Terms

API, Time Series Analysis, Wikipedia

## 1. INTRODUCTION

In recent years many so-called *zeitgeist* applications have been launched. Such applications are used to gain insights into the current gist of society and actual affairs. Several news sources run zeitgeist applications for popular and trending news.[1] In addition, there are zeitgeist applications that report on trending publications such as LibraryThing,[2] and trending topics, such as Google Zeitgeist.[3] In web dynamics, users' visitation and search patterns were modeled [1, 6].

Most zeitgeist tools are based on proprietary data and it is therefore often impossible to see beyond the limited amount of information shown by the tools. For many scenarios, though, it would be useful to be able to gain insights in people's interests. Three examples of time-aware information access scenarios in which such a need arises naturally are described below.

**Data mining expert.** A data mining expert wants to do experiments based on the popularity of Wikipedia concepts over time. He is a financial analyst and would like to know if there are correlations between stock market rates and the general interest for a company. As a financial analyst, he has his own time series analysis tools for financial data, but misses temporal data that captures "general interest" and needs access to the raw data of a few (<20) Wikipedia concepts, including the company and its products.

**Brand analyst.** A brand expert would like to analyze the current interest, i.e., zeitgeist, regarding her brand. She is interested in her own brand, but also other brands, related to hers. But what does related mean? For her, this means brands that share trends, bursts, and recurring events. She would like to have the results presented visually, in a graph, but also be able to create her own statistics based on machine readable results.

**Wikipedia moderator.** Wikipedia has moderators, responsible for the evaluation and monitoring of a certain set of articles. They are interested in how many times an article is viewed, in order to distribute valuable and sparse volunteer work to articles of interest. The moderator asks if a certain article receives more interest than it used to. He is interested if one article in a collection of articles (a category) experiences sudden interest, or cluster articles with similar trends of interests into groups. He is not interested in the full set of page view counts per concept, but a subset of similar concepts. He is not interested in raw data, but a bird's eye view that reveals trends.

There is an interesting open data source from which a stream of people's changing interests can be observed across a very broad spectrum of areas: the Wikimedia access logs.[4] The logs contain the number of requests made to any Wikimedia domain, sorted by subdomain and aggregated on an hourly basis. Since they are a log of the actual requests, they are noisy and can also contain non-existing web pages. They are also quite large, yielding 60 GB worth of compressed textual data per month. Currently, we update the data on a daily basis and filter the raw source data by matching the URLs of all English Wikipedia articles and their redirects.

In the remainder of this paper we describe an API that facilitates easy access to the access logs. The API works as an interface for three types of user, the data mining expert, a brand analyst, and a Wikipedia moderator, as described above. From these use cases we have identified the following requirements our system should have:

- The user must have access to the raw time series data for a concept.
- The user must be able to find the *N* most temporally similar concepts.
- The user must be able to group concepts and their data, based

[1] For example, the The Japan Times runs a feature titled "The Zeit Gist" every Tuesday, which covers community issues particularly relevant to the foreign community within Japan.
[2] http://www.librarything.com/zeitgeist
[3] http://www.google.com/intl/en/press/zeitgeist2010/
[4] http://dumps.wikimedia.org/other/pagecounts-raw/

```
http://opengeist.org/{begindate}/{enddate}
       *{resolution}/*{filter}
              [/category/{id}
                     [/raw]
                     [/topN[/compare/*{filter_graph}]
                     [/cluster[/k/*{filter_graph}]]
                   ]
                     [/statistics]
              ]
             [/concept/{id}
                     [/raw]
                     [/topN
                            [/cluster[/k/*{filter_graph}]]
                            [/compare/*{filter_graph}]
                     ]
                     [/statistics]
              ]
              [/statistics]
```

Figure 1: Syntax of the GET commands for the RESTful webservice of OpenGeist: terms in `{}` are variables and if marked with `*` they are optional.

either on the categorial system of Wikipedia or on similarity between concepts.

- The system must return either a textual or a visual representation.
- The user should be able to apply time series filters to extract trends and (recurring) events.

We are aware of several existing solutions to accessing the Wikipedia page view information. Ciglan and Nørvåg [2] present a recommender system for new and popular articles, based on favorited Wikipedia articles and page views. There are two systems that display statistics about Wikipedia pages: Wikistics[5] shows top viewed articles and search terms, as well as time charts for 2009, and the Wikipedia user `aka` provides a tool for monthly edit statistics. Both tools lack a REST interface and means to find pages with *similar* page view or edit history.[6] The Wikipedia user `Henrik` provides a REST interface[7] to access the raw time series data but no filtering or comparative graphs of similar pages.

In the remainder of the paper, we describe a REST interface that provides the raw data for a concept as well as different approaches to data aggregation methods (Section 2). In Section 3 we give two example applications and we end with a concluding section.

## 2. API

In this section we describe an API for a RESTful webservice [3, chap. 5] that allows users to retrieve the Wikipedia access logs data in either raw, aggregated, or filtered form. It allows users to cluster different concepts according to user interest or look at different aggregations of the data.

Figure 1 describes the full syntax of the GET requests that define the API. The API is safe, as we provide only GET methods. The *id* is a uniform resource identifier, but is different from the URI as used for Wikipedia concepts: instead of the full URI we use the title of a page. We distinguish between *concept* and *category*, where a concept can be a member of a category. A category itself can be a concept and therefore a member of a (different) category.

[5] http://wikistics.falsikon.de/long/wikipedia/en/
[6] http://vs.aka-online.de/cgi-bin/wppagehiststat.pl
[7] http://stats.grok.se/about

To avoid cycles, we say that a concept is only a member of its immediate parent category. Requesting a category as category page implies the need for aggregation, while requesting it as a concept entails that the user wants information about the page itself. The use of Wikipedia titles is natural as it forms a direct link to the DBpedia knowledge graph and therefore to linking open data. For every request we provide two representations: (i) in json format for further processing and with metadata and (ii) in image format (png) for viewing in a browser. The latter is requested by adding `.png` at the end of the GET request.

The temporal *resolution* of the time series and *filter* are to be set optional, the default resolution is a day and the data is by default unfiltered. The final, also optionally, filter argument, *filter_graph* is for the visual representation: if a filter is set, the time series for the visual representation is preprocessed based on this filter, otherwise it plots the raw data. A time series as the API returns it, is the number of page views per time unit. A time unit can be a day, week, or a month.

The remainder of this section we give examples of GET requests based on the use cases introduced in Section 1. In Section 2.1 we describe the basic retrieval of time series, and in Section 2.2 how to get general collection statistics. Section 2.3 describes how data can be grouped.

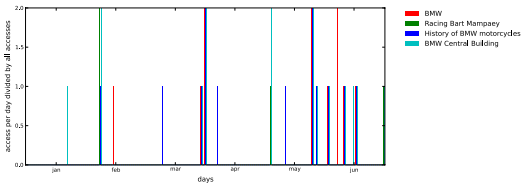


Figure 2: The four concepts closest to `BMW` using the `burst`.

### 2.1 Raw series

For people with expert knowledge the raw data is desirable. The user needs to be able to directly access the interface and retrieve page views per day directly without filtering. For example, the expert would like to get the time series for the company `BMW`, the call of the API would be:

`http://opengeist.org/20110101/20111231/concept/BMW/raw.`

The result is the raw data in json format:

```
{   'id':'BMW',
    'timeseries': {'2011-01-01':12, ... , '2011-12-31': 20}}.
```

This object contains the id and a time series object. This interaction is similar to the interface provided by http://stats.grok.se/. Following the earlier mentioned design principles, the request

`http://opengeist.org/20110101/20111231/concept/BMW/raw.png`

returns the time series as a png, and the request

`http://opengeist.org/20110101/20111231/category/German_brands/raw`

returns all concept objects with time series of the concepts in the category of German brands:

```
{'category':'German_brands',
 'concepts':
      {'id':'BMW',
       'timeseries': {'2011-01-01':12,...,'2011-12-31': 20}}...}.
```

The visual representation is a comparative plot of the page views per day.

## 2.2 Statistics

The brand analyst is preparing a report for the public relations team of BMW. In order to report the knowledge and interest of users concerning the brand, she requests

```
http://opengeist.org/20110101/20111231/concept/BMW/statistics
```

which will return a json file with the mean, standard deviation, maximum, and minimum of the page views.

Similarly, the Wikipedia moderator, trying to assess the importance of his work, requests the statistics for a category with

```
http://opengeist.org/20110101/20111231/category
/German_Brands/statistics.
```

This returns the aggregated statistics of all concepts in the category as well as a statistics object for each concept.

The global collection statistics can be accessed with

```
http://opengeist.org/20110101/20111231/statistics.
```

This shows number of pages, categories and overall views per day.

## 2.3 Grouping

The brand analyst needs to compare time series of concepts. For that, we provide an interface that allows the user to select the most similar concepts, but also cluster similar concepts and return typical concepts (prototypes) of the cluster.

An example request that returns similar concepts is

```
http://opengeist.org/20110101/20111231/month/trends
/concept/BMW/3/compare.
```

Here, `trends` is the filter (see below) and the request returns the three concepts with the most similar time series:

```
{'id':'BMW',
  'timeseries': {'2011-01':80, ... , '2011-12': 20},
  'timeseries_filtered': {'2011-01':60, ... , '2011-12': 15},
  'similar_concepts': [
    { 'id':'Racing_Bart_Mampaey',
      'timeseries': {'2011-01':40, ... , '2011-12': 30},
      'timeseries_filtered': {'2011-01':60, ... , '2011-12': 34},
      'distance' = 34.49 },
    { 'id':'History_of_BMW_motorcycles', ... }... ]  }
```

using the canberra distance metric [4].

The filter indicates in which respect the time series are similar: it preprocesses the time series and extracts information relevant for the specific task. It can request a moving average filter using either `trends` or `moving_average`, it can request significantly different viewing patters using `burst`, or it can request regular patterns with either `recurring` or `fourier`. Using the filter `trends` or `moving_average` applies a moving average linear filter, with a window size of 10. It eliminates regular patterns (e.g. more views on weekends) and keeps track of a general trend of the distribution. The filter `burst` sets all days with page view counts bigger than two standard deviations of the mean with two, proceeds similarly for one standard deviation, and sets the rest to zero. This identifies months with a high number of accesses, and disrespects trends and regular patterns. The filter `recurring` and `fourier` is the superposition of the dominant fourier curves of the time series. This filter emphasizes regular events and patterns in the views of Wikipedia pages. The visual representation will be a graph overlaying the topN (here three) similar curves. Figure 2 is the result of the call:

```
http://opengeist.org/20110101/20111231/month/burst
/concept/BMW/4/compare/burst.png
```

where the similarity is calculated based on the burst filter and the visual representation are the burst-filtered time series.

### 2.3.1 *Clustering of concepts*

For the brand analyst, the most similar brands are interesting to be compared with either direct competitors or related brands. Clustering concepts similar to a concept into groups allows the analyst to identify sources of temporal similarity.

A request for concept clusters with similar time series for the brand BMW is

```
http://opengeist.org/20110101/20110630/bursts/concept
/BMW/20/cluster/4.
```

The system returns the four prototypes of the cluster, the cluster assignments and statistics:

```
{'id':'BMW',
  'timeseries':
     {'2011-01-01':12, ... , '2011-12-31': 20},
  'timeseries_filtered':
     {'2011-01-01':1, ... , '2011-06-30': 2},
  'clusters':
     [{'id' : 1,
         'prototype' :
           {'timeseries':
             {'2011-01-01':0, ... , '2011-06-30': 2},
                'distance' : 30
             },
         'assignment':[
           {'id':'BMW_1_Series',
              'timeseries':
                {'2011-01-01':8, ... , '2011-06-30': 12},
                   'timeseries_filtered':
                     {'2011-01-01':0, ... , '2011-06-30': 2},
                        'distance' = 2
  }...] ...} ... ]  }.
```

The cluster algorithm used is $k$-means, with a cluster size $k = 4$ (default $k = 5$). The returned cluster statistics are the size, the width and the variances of a cluster. Following the same design principles as the grouping, the clustering can be done on different filters and uses the canberra distance matrix. If a visual representation is requested, the system returns a comparative plot of the cluster prototypes with an attached list of cluster assignments, sorted by distance to the prototype. This allows the brand analyst to associate similar brands with certain events. The visual representation is in Figure 3.

### 2.3.2 *Clustering of categories*

The Wikipedia moderator wants to focus on articles that receive increasingly more attention. Instead of looking at every single time series of an article independently, he can cluster trends of articles and select the cluster with a trend to more page views. His GET request looks like

```
http://opengeist.org/20110101/20111231/day/trends/category
/German_brands/cluster.png.
```

The returned graph is similar to Figure 3, but based on the members of the category `German_Brands`.

# 3. EXAMPLE APPLICATIONS

In this section we describe two applications that make use of the Wikimedia access logs data.

**Semantic Linking.** Meij et al. [5] link tweets to Wikipedia concepts. This linking enriches the microblog posts by providing a contextualization of the information. Due to their unedited and noisy nature, semantic enrichment enables many new applications, ranging from retrieval to reputation monitoring. The authors use a broad range of features in a machine learning context, including those based on Wikimedia access logs (WIKISTATS and WIKISTATSWK). WIKISTATS is the frequency with which the page was

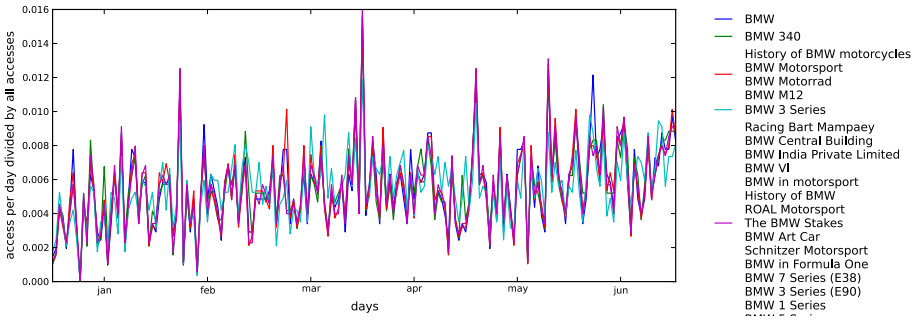


Figure 3: The time series for the page views of the concept `BMW` in blue. Clustering the 20 closest concepts to `BMW`, results in four prototypical time series in green, red, cyan and magenta.

visited in an entire year and WIKISTATSWK the frequency with which it was visited in the seven days before the tweet was posted. Instead of processing the entire dataset, the authors can now process the features on demand by requesting the concept statistics over the entire year,

```
http://opengeist.org/20100101/20101231/entity
/Judy_Dench/statistics
```

over the last week before a tweet was posted,

```
http://opengeist.org/20100620/20100627/entity
/Judy_Dench/statistics
```

and the general collection statistics for normalization:

```
http://opengeist.org/20100101/20101231/statistics.
```

**Reputation monitoring system.** The number of page views for an entity are an indicator of its fame and general public interest. But what drives a sudden interest for an entity? With our technology, one can easily build a reputation management system that monitors different aspects of an entity and links them to a textually richer reflection of public interest: Twitter. The reputation management system is a dashboard with a monitoring, an alerting, and an informing component. The monitoring component shows page views of pages related to the entity in question: as can be seen for the example BMW, there is more than one Wikipedia page related to BMW, all of them subtopics. If there is a sudden change of page views for one of the subtopics, the alerting component warns the user. If desired, the informing component shows tweets that are about this subtopic.

The API helps with each of the aspects in the monitoring system. It helps selecting temporally related concepts for an entity; for the example entity BMW it can select Wikipedia pages similar to the concepts in Figure 3. For the visualization and monitoring of the entity, it returns a plot for the page views for every single entity. If there are more page views than usual, the filter option `bursts` filters such recent (and general) increases. Finally, the linking of tweets to an article can be done using methods from Section 3.

# 4. CONCLUSION

We presented a RESTful web interface to request and retrieve time series for Wikipedia concepts. The time series are based on the logs of the access counts of the Wikipedia page of a concept. This interface does not only return the raw time series, but also filters time series and clusters temporally similar concepts, returning visual or textual objects. The requirements were created based on three use cases, based on technical novices as well as an expert and we showed how the API can suit their needs. For future work, we intend to use this API for experiments and create dashboard interfaces for different user bases. This will certainly result in an expansion of the interface.

**Acknowledgments.** This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.-004.802, 380-70-011, 727.011.005, 612.001.116, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP and BILAND projects funded by the CLARIN-nl program, the Dutch national program COMMIT, and by the ESF Research Network Program ELIAS.

# 5. REFERENCES

[1] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW 2005*, 2005.

[2] M. Ciglan and K. Nørvåg. Wikipop: personalized event detection system based on wikipedia page view statistics. In *CIKM 2010*, 2010.

[3] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[4] G. N. Lance and W. T. Williams. Mixed-Data Classificatory Programs II - Divisive Systems. *Australian Computer Journal*, 1(2):82–85, 1968.

[5] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM 2012*, Seattle, 2012. ACM.

[6] K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Modeling and predicting behavioral dynamics on the web. In *WWW 2012*, 2012.