Real-time Aggregation of Wikipedia Data for Visual Analytics

Nadia Boukhelifa*
Microsoft Research - INRIA Joint Centre

Fanny Chevalier†
Microsoft Research - INRIA Joint Centre

Jean-Daniel Fekete‡
INRIA

ABSTRACT

Wikipedia has been built to gather encyclopedic knowledge using a collaborative social process that has proved its effectiveness. However, the workload required for raising the quality and increasing the coverage of Wikipedia is exhausting the community. Based on several participatory design sessions with active Wikipedia contributors (a.k.a. Wikipedians), we have collected a set of measures related to Wikipedia activity that, if available and visualized effectively, could spare a lot of monitoring time to these Wikipedians, allowing them to focus on quality and coverage of Wikipedia instead of spending their time navigating heavily to track vandals and copyright infringements.

However, most of these measures cannot be computed on the fly using the available Wikipedia API. Therefore, we have designed an open architecture called *WikiReactive* to compute incrementally and maintain several aggregated measures on the French Wikipedia. This aggregated data is available as a Web Service and can be used to overlay information on Wikipedia articles through Wikipedia Skins or for new services for Wikipedians or people studying Wikipedia. This article describes the architecture, its performance and some of its uses.

Index Terms: Information Storage and Retrieval [H.3.5]: Online Information Services—Web-based services; Database Management [H.2.1]: Logical Design—Schema and subschema; Database Management [H.2.4]: System—Query processing

1 INTRODUCTION

Wikipedia is well established as an extensive repository of encyclopedic knowledge available for free on the Internet.

Gathering that knowledge is done by an active community of volunteers (Wikipedians) who act in several ways. Based on the *Reader-to-Leader framework* [15] and working sessions with Wikipedians, we have identified six tasks specific to Wikipedians:

(T1) writing some of the articles,

(T2) fixing multiple issues with existing articles,

(T3) advising new contributors on how to edit articles,

(T4) animating thematic groups (WikiProjects) and planning for global improvements of articles related to these groups,

(T5) watching articles to prevent them from vandalism,

(T6) resolving social issues, disagreements or "edit wars".

To effectively improve the encyclopedia, Wikipedians should devote most of their time to T1–T4 when they actually spend most of their time on items T5 or T6. As HCI and Information Visualization practitioners, how can we help improve this situation?

*e-mail:Nadia.Boukhelifa@inria.fr
†e-mail:Fanny.Chevalier@inria.fr
‡e-mail:Jean-Daniel.Fekete@inria.fr

IEEE Symposium on Visual Analytics Science and Technology
October 24 - 28, Salt Lake City, Utah, USA
978-1-4244-9486-6/10/$26.00 ©2010 IEEE

To answer this question, we have organized several participatory design sessions with Wikipedians and gathered information about their way of working, how they spend their time and what kind of tools would help them carry out their tasks. We report the results of these sessions in Section 4.

From these sessions, we extracted a set of computable measures that can help Wikipedians decide faster on actions to take when monitoring changes to articles. For example, to quickly assess the trustworthiness of an unknown contributor, they could look at the tag-cloud of the user's contributions or at a well-established measure of trust [1]. These two mechanisms would be effective but are not yet available on Wikipedia. Computing them is expensive in time and resources and the Wikipedia foundation is already spending all its efforts maintaining the Wikipedia infrastructure. Therefore, despite the efforts made by the Machine Learning and Information Visualization communities to describe how to compute and present effective aggregated information on Wikipedia, this information is still not available to Wikipedians.

To overcome this shortcoming, we have designed the *WikiReactive* architecture to compute and maintain aggregated information on the French Wikipedia. This architecture extracts information continuously from the main Wikipedia server using its standard Web-based API and aggregates it in real-time.

We first give some background on the actual infrastructure of Wikipedia. After the related work section, we report the results of several participatory design sessions we organized with Wikipedians to clarify their requirements. We then describe the *WikiReactive* generic architecture and the specific information we currently maintain along with the deployed algorithms and measures of performance. The following section presents some of the visualizations we have connected to our *WikiReactive* server before concluding.

2 THE WIKIPEDIA INFRASTRUCTURE

Wikipedia is an Encyclopedia service provided by the Wikimedia Foundation, a nonprofit charitable organization. As of April 2010, it is available in 240 languages. The English Wikipedia is the most complete and contains roughly 3 million articles; the German and French Wikipedia contain about 1 million each.

Wikipedia uses the MediaWiki[1] Wiki system relying on a database to manage about 40 tables[2]. An article in Wikipedia is managed as an ordered list of revisions that are stored as full text encoded in WikiText, the internal format used for editing articles. MediaWiki uses one database per language and, from a technical point of view, each language is managed as a distinct service. Wikipedia servers are also replicated worldwide.

MediaWiki provides a Web-based interface for browsing and editing articles; this is what readers see. It also provides a Web-based API[3] for programs to access Wikipedia data. The API is similar to accessing the database but it limits the chances of abuse. Wikipedians access Wikipedia through the Web interface and several of them also use Robots (a.k.a bots) to perform automatic tasks[4]. These bots can be used for collecting information, processing data continuously (e.g. update infoboxes, check for copyright

[1] http://www.mediawiki.org/wiki/MediaWiki
[2] http://www.mediawiki.org/wiki/Manual:Database_layout
[3] http://www.mediawiki.org/wiki/API
[4] http://en.wikipedia.org/wiki/Wikipedia:Bots

violation) or editing articles. For example, the bot called *SineBot* runs continuously and adds a signature next to an unsigned comment left in a *Talk* page. Several bots collect statistical information that is made available on external sites.

For more sophisticated extensions, the Wikimedia Foundation provides the *Toolserver*: "a collaborative platform providing Unix hosting for various software tools written and used by Wikimedia editors"[5]. The Toolserver database contains a replica of all Wikimedia wiki databases. Software running on the Toolserver can access the databases directly and perform queries that would be too expensive using the API. The Toolserver, thus, allows for faster and non-restrictive access to the data, but it suffers from replication lag ranging from seconds to about 20 minutes[6].

Most of the software applications that rely on the Toolserver are bots that scan the whole Wikipedia in search of interesting information or changes to apply automatically. Accounts on the Toolserver are granted by a committee according to a set of rules. Once the account is granted, the bots and database queries also need to be approved by a committee before they are allowed to run in order to avoid stalling the servers.

A service to aggregate Wikipedia information can therefore be implemented at several levels: at the *page level* by scraping Wikipedia pages, at the *API level* using the API to get information, or at the *service provider level* by adding programs on the Toolserver to access the database. The API level is very easy to use but limited. The Toolserver is very powerful but provides data that is not always up-to-date and also requires going through several administrative processes.

3 Related Work

The success and the popularity of Wikipedia have raised interest from Machine Learning and Information Visualization communities. Thus, several studies aiming at understanding and visualizing the editing activity in Wikipedia have been proposed.

3.1 Measures on Wikipedia

Most of the activity in Wikipedia needs to be devoted to improving the quality of the encyclopedia, but its fast changing and volatile content makes "the free encyclopedia that anyone can edit" prone to unverified information and conflicts. As a consequence, Wikipedians spend a lot of their time patrolling. There have been several studies meant to help improve their awareness of the editing activity and reduce the time they devote on patrolling.

The qualitative studies attempt to assess the quality of Wikipedia articles in an objective way. They rely on metrics based on the meta-data associated with Wikipedia itself which can be separated into two categories: the *per-article* metrics and the *per-user* metrics. Among the per-article metrics, the word count [2], the number of edits and the number of unique contributors [13, 22] have been used as measures of quality. The word count revealing the "completeness" of an article, the number of unique authors its "diversity" and the number of editions its "rigor". In the same category, and taking into account the lifecycle of an article, changes in transient and persistent contributions have also been used for measuring article quality [23]. An example of a per-user metric is the "author reputation" introduced in [1] as the amount of remaining text in the final version of the articles the author has contributed to divided by the amount of text entered by this same author. Overall, these user-oriented metrics are cautiously used since measuring the reliability of a contributor is a sensitive question – much advice can be found on Wikipedia to avoid the so-called *editcountitis* phenomenon that is related to measuring the quality of contributors by their edit count. Other types of metrics based on the aggregation of several indicators have been proposed for predicting the quality of user contributions [8] or the trustworthiness and the quality of an article [7, 12].

This data is not only useful to apprehend the activity both from the users and the articles point of view, but as these studies reveal, social information can also be used as quality indicators for assessing the content of Wikipedia. This can then help the community to improve the encyclopedia. It has been shown that revealing trust-relevant information to the user has an effect on the perceived trustworthiness of the articles [11, 14]. Indeed, a couple of visual tools have been proposed aiming at enhancing the user and reader experience with Wikipedia.

3.2 Visualizations for Wikipedia

Both the History Flow visualization [19] and WikiDashboard [14, 18] show the evolution of an article over time. WikiDashboard provides an *article dashboard* that shows the weekly edit activity of the article, followed by a list of the corresponding main authors' activities; and the *user dashboard*, that displays the global weekly edit activity of the user, followed by a list of the pages the user has edited the most. WikiDashboard uses data directly available through the Wikipedia Toolserver that can be fetched in almost real-time from the Wikipedia database. It does not use aggregated information but its visualizations provide some degree of visual aggregation. The History Flow visualization relates the length of an article to the number of changes (characters added, removed or moved) and their authors. It needs to fetch its data from Wikipedia and compute the diffs in the user's machine memory, which requires a substantial amount of time for active articles, e.g. up to 30 minutes to retrieve all the revisions of the article *France*.

Two systems provide aggregated information in the context of Wikipedia pages: WikiTrust[7] [3] and WikipediaViz [5]. WikiTrust computes an index of text trust, which is a measure of how much the text has been revised by reliable users. This trust is encoded via background text coloring using a white-orange heatmap; white background corresponding to "trusted text" — high-reputation authors have seen the page since the text has been inserted and have left the text unchanged —, and dark orange background highlights "untrusted text" — text that has been inserted, or modified, recently. WikipediaViz has been designed to help casual Wikipedia users assess the quality of articles. It implements five thumbnail visualizations, each conveying a metric associated with the article including the word count, the contributors and their contributions, the article timeline, the internal links and the discussion activity. Both of WikiTrust and WikipediaViz rely on aggregated data that requires the computation of the diff across the different revisions. In their current state, they both provide information computed on a static version of Wikipedia and, therefore, are of little use to Wikipedians who want up-to-date information, as accurate as possible.

Recently, *iChase* [16], an interactive multi-scale visualization of the editing activity of a WikiProject, has been proposed to help Wikipedians monitor their projects. *iChase* provides a dual-entity exploration for both articles and contributors in a unique visualization. Even though it visually conveys aggregated information on its timeline-based visualization by the use of heatmaps, iChase relies on detailed meta-data gathered through the API. It does not require the actual content of the articles making it faster to gather the data compared to other tools.

3.3 Wikipedia statistics tools

There exist numerous tools providing statistics on Wikipedia that can be classified into two categories: global and local statistics. The global statistics tools[8] show the evolution of simple aggregated data on the whole Wikipedia such as article traffic, article count, revision

[5] http://www.toolserver.org/

[6] Replication lag statistics at http://toolserver.org/~bryan/stats/replag/

[7] http://www.wikitrust.net/

[8] http://en.wikipedia.org/wiki/Wikipedia:Statistics

count, and user count. They display the data at a daily rate but do not require accurate real-time data. The local statistic tools[9] focus on articles or users. They show either user-centric aggregated data, such as the number of edits of a user and the list of the most edited articles; or article-centric aggregated data, such as the number of distinct contributors or the most frequent contributors. They require computations on the fly that are time consuming (e.g. 10 seconds to about one minute in WikiChecker[10]). Furthermore, they usually rely on the Toolserver database for a quick and non-limited access to the data and therefore do not guarantee up-to-date information.

3.4 Data warehousing

The problem of maintaining aggregated data in near real-time is not specific to Wikipedia. For instance, in software management, it is common to keep track of hundreds of files in a version control system, each version having thousands of revisions. In this context, the computation of aggregated data is relevant for software evolution visualization [20]. On-the-fly computation of aggregated data remains possible since the data is not as volatile as Wikipedia — Wikipedia is an extreme case in that it deals with a large amount of articles that are evolving quickly in time. However, as pointed out in [20], the availability of a robust, efficient, well-documented, usable mechanism to query a software configuration management repository is not always guaranteed.

The problem of correlating heterogeneous data has to be mentioned as well. For instance, *Hipikat* [6] and *softChange* [9] aggregate different related sources such as bugzilla, CVS repository, emails, etc. to correlate software trails from different sources and infer relationships between them. In this specific case, maintaining aggregated data aims at grouping heterogeneous data within a common high-level concept. In such a context, having a real-time continuous data storage might not be the primary focus.

On the other hand, workflow management systems are confronted with the same problems of managing continuous data [17]. However, a mechanism for propagating data with minimal latency into a data warehouse remains to be done for Wikipedia.

As a conclusion, there has been much effort by the Machine Learning experts, Information Visualization practitioners and Wikipedians to provide users with relevant metrics on Wikipedia. However, the existing tools either rely on live but detailed data; or they propose to visualize aggregated data that is not up to date — and, thus, of little interest to Wikipedians —; or they offer off-line solutions because of the cost of heavy computations.

4 Participatory Design with Wikipedians

To better understand the needs of Wikipedians, we organized 4 participatory design sessions with a total of 20 different heavy contributors and administrators. The sessions were organized as full days, during which we collected scenarios and discussed current practices of the participants, introduced them to a wide variety of visualizations and other tools, and organized brainstorming sessions and video prototyping of their ideal system.

4.1 Wikipedian practices and needs

During the participatory design sessions, we focused on the work process Wikipedians follow on a daily basis and how to improve it. They acknowledged the results of the study of Wattenberg and Viégas [21] on the diversity of Wikipedians and explained that they had very different working habits, styles and interests. At the highest level, all of them spend time managing the articles they monitor. This is the most tedious task they have to achieve because it can take an unbounded amount of time, though this is necessary to maintain the quality of Wikipedia.

[9] http://meta.wikimedia.org/wiki/Toolserver/Projects
[10] http://en.wikichecker.com

4.1.1 Article-centric approach.

To stay informed about article activity, Wikipedians use the Wikipedia "watchlist" mechanism: each registered user can add articles on their watchlist. The Wikipedia interface currently provides a history of all the changes that occurred to that set of articles in chronological order as a textual list; one line per revision containing the article title, the date and time of the change, the number of characters added or removed overall, the name of the user (or IP number for anonymous changes) and a short comment. Then, Wikipedians have to read the revisions and rely on their knowledge to decide what to check and in what order.

4.1.2 User-centric approach.

Assessing the quality of an unknown contributor is one of the most time-consuming sub-tasks. This can be done from the actual contribution, from the profile of past contributions and from interactions with other Wikipedians, available on the user's personal page or on other pages that are not available centrally. Thus, when looking at the contribution of an unknown user, Wikipedians have to navigate to hunt for information and make a decision: they can keep the contribution, edit it or revert it. Even when they have decided what to do, the work is not done. If the contribution is a vandalism, they track back the vandals and revert all their changes. This operation is tedious using the current Wikipedia interface. If the contribution needs editing, Wikipedians could advise newcomers on how to contribute correctly. This again is time-consuming because the level of advice will depend on the profile of the user: a novice user should be initiated with some rules and guideline practices in a welcoming way whereas frequent contributors who do not comply with the rules should be warned. Currently, finding-out the profile of a contributor in terms of activity (so that the appropriate tone is used when advising), or in terms of interests (in order to encourage users to contribute to specific articles) requires tedious navigation.

4.1.3 Group-centric approach.

Some Wikipedians who also belong to a WikiProject are meant to coordinate the Wikipedia activity around a topic. The level of coordination and involvement of Wikipedians in a WikiProject varies greatly depending on the people, the projects and the language. However, the Wikipedia community acknowledges that WikiProjects and coordination are important for reaching higher quality and consistency in Wikipedia. As far as we know, only iChase [16] provides awareness of activity on a project but it is not yet available for download. Currently, assessing the activity of a group requires explicitly browsing the group's page on Wikipedia.

4.1.4 Quality measures.

As the Wikipedians spend most of their time dealing with quality issues, they all asked for tools to raise awareness of quality issues for them as readers. The session resulted in two kinds of methods to solve this problem: objective and subjective measures. Objective methods expose objective information on the main article page whereas subjective solutions try to find aggregated scores to provide a quality rank of the page. Although both methods seem useful, we focus on objective methods because we feel they are more in the spirit of Wikipedia (neutral, verifiable and factual). Most of the criteria expressed by expert participants were similar to the metrics already presented in studies about Wikipedia, which we have mentioned in Section 3.1.

In the end, Wikipedians take decisions that trade time spent on monitoring — which can be unbounded — in for time spent on adding of improving new content. They all asked for better interfaces to quickly assess the quality of contributions and contributors, and to facilitate tasks. Providing these interfaces needs aggregated information. This is not available through the current Wikipedia interface and takes too long to compute on demand.

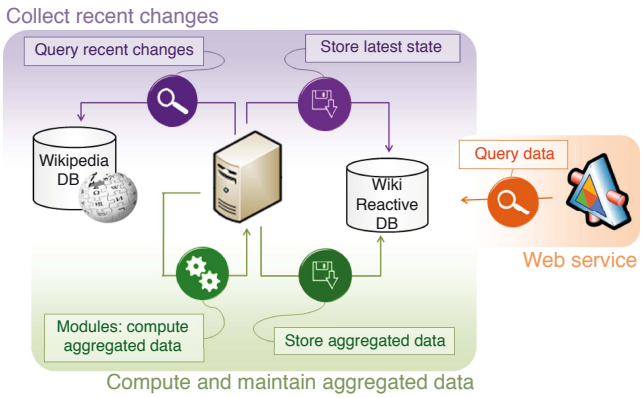

Figure 1: Overall scheme of the *WikiReactive* infrastructure

This is why we have implemented *WikiReactive*: an open architecture to compute important aggregated information for on-demand delivery to Wikipedians in respect with a set of goals listed in the following section.

4.2 Design Goals

The large number of collected scenarios and our discussions with Wikipedians confirm the difficulties they encounter to efficiently achieve tasks for motivating social participation (T3-T4). From the tasks and design goals listed in [16], the working sessions and the technical requirements, we derived a set of 4 general goals for *WikiReactive*.

(G1) Facilitating the awareness of articles' activity and status: Wikipedians need to monitor the progress of articles to identify those requiring more effort in detecting unusual activities.

(G2) Facilitating the awareness of contributors' activity and profile: project participants need to be aware of contributors' interests to encourage them to take part in specific articles or mentor new contributors. They also need to monitor the activity of new contributors or vandals.

(G3) The maintained data has to be neutral, verifiable and factual. To remain in the same spirit of Wikipedia, i.e. encouraging objectivity and neutrality, the aggregated data has to rely on objective metrics.

(G4) Incremental and atomic computations to guarantee the currency of the aggregated data and prevent problems that may occur as a result of server disruptions: the infrastructure needs to be robust.

Despite the importance of group-centric awareness, we decided not to consider it in this first version of *WikiReactive*.

5 Design of the Wikipedia Aggregation Server

The overall infrastructure of *WikiReactive*, depicted in Figure 1, is made of three parts:

- **Collecting Recent Changes:** A mechanism for collecting and storing recent changes in a table;
- **Modules Cascade:** Several modules to compute and maintain aggregated information in multiple tables;
- **Web Service:** A Web Service to allow external applications to access the data.

This section describes the three parts and reports on the results in terms of performance for computing and serving the data.

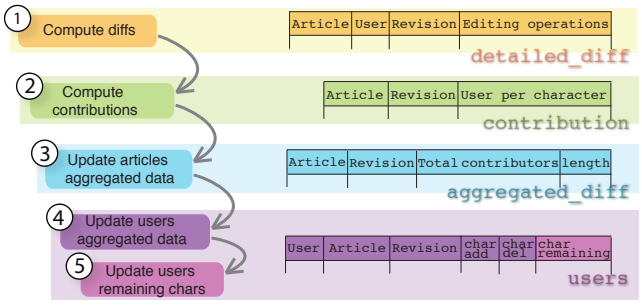

Figure 2: Diagram of the modules cascade in *WikiReactive*.

5.1 Collecting Pages to Process

The modules cascade relies on the latest *state* table to determine the new articles or the newer revisions to process. To maintain that table, we use a polling mechanism to continuously query the Wikipedia API and collect the *Recent Changes*: we ask for all the changes since the last time we processed the revisions. When the list of changes is less than a 100 items, we delay the next query for 4 seconds. The API gives details about the articles that have been changed: what happened (new revision, new article, article removed, revision reverted, etc.), the user who made the change, the time-stamp of the change and a few details. We update the state table by creating the article if needed; we associate its newest revision and the corresponding time-stamp.

Then, we perform the cascade to disseminate the new event data by activating triggers for update on all the other modules.

This modular architecture is designed to facilitate the addition of extensions and new services without requiring the *WikiReactive* service to stop the modules. Each extension needs to create a set of tables, install triggers on existing tables if needed and start polling for changes.

5.2 Goal of the Cascade: Merging WikiTrust and WikipediaViz

WikiTrust [3] and WikipediaViz [5] both visualize objective aggregated metrics that require the processing of the whole articles history (see Section 3.2). To perform that computation incrementally, we have implemented the cascade of the five following modules (Figure 2). We describe them in details in the next section.

1. For each article, compute and maintain the differences (diffs) between successive revisions;
2. For the last revision of each article, compute and maintain the *contribution* table that stores at each character index the identifier of the user who entered it;
3. For each article, compute the number of distinct contributors;
4. For each user, maintain the total number of characters entered per article;
5. For each user, maintain the total number of characters remaining per article.

5.3 Modules Cascade

5.3.1 Module 1: Diff Computation.

Wikipedia only provides the full text of each revision so the changes made by a contributor from one revision to the next have to be computed. We use a variant of [10] that computes a minimal set of editing operations (insertions, deletions and moves) at the character level to mimic the edition process. This produces a good guess of the contributor's changes.

For each *revision*, we store the detailed list of *editing operations*[11] as well as the associated *article* and *user* in the *detailed diff* table as shown in the following example.

Article	User	Revision	Editing operations
1290	2	6738	insert(0, 9, "some text")
1290	7	6739	insert(4, 6, "random"), delete(0, 4)

In the previous example, the user 2 has created the article number 1290, and its content: "some text" (revision 6738). Then, the user 7 has modified it (revision 6739) to end up with "random text" as the content (insertion of the word "random" at index 4, then deletion of 4 characters, starting from index 0).

5.3.2 Module 2: Per-User Contribution Computation.

This step consists of assigning each user an identifier and using it to sign each character in the revision. Users are then referenced by id to facilitate indexing and database joins[12]. The implementation considers the contribution table as a string made of user identifiers. We start with an empty string and apply the diff operations computed at the previous step, using the user id instead of the string contents as illustrated in the following example (we keep the same example of the two revisions made on the article 1290).

s	o	m	e		t	e	x	t
2	2	2	2	2	2	2	2	2

r	a	n	d	o	m		t	e	x	t
7	7	7	7	7	7	2	2	2	2	2

The change operations (insertion, deletion and move) are performed on the user's id instead of the user's text.

Currently, we only store the *contribution* table of the *latest revision* of each *article* (that is "77777722222" for the article 1290, revision 6739).

5.3.3 Module 3: Article Activity Computation.

Besides the diff, we also store the sum of characters touched by the different change operations in the *aggregated diff* table for each *revision*. This information allows tools like WikipediaViz [5] to show the evolution over time of the activity of an article per revision or time, highlighting the stabilization of an activity: a trail of several revisions with minor changes.

Article	Revision	Total Contributors	Length
1290	6738	1	9
1290	6739	2	11

5.3.4 Module 4: User Contribution Computation.

This module consists of keeping in the *users* table the aggregated sum of users operations on individual articles for each user. This is done incrementally after the article activity has been computed. This step is a cache for an expensive SQL sum query.

Note that in this table, the pair `(user, article)` is used as the primary key. As a consequence, we only keep the aggregated values that correspond to the latest time a user has touched the article (see the example in the following paragraph).

[11]The detailed operations are: insert(from, length, text); delete(from, length) and move(from, length, to)

[12]Note that we maintain an extra table for users to assign an auto-incremented id to each name

5.3.5 Module 5: Character per User Computation.

This last module consists of updating, for each user, the number of characters remaining in each articles he or she has contributed to. This is done by counting the number of user ids occurring in the contribution table and storing it in the *remaining char* column of the *users* table.

Note that one user's changes might affect the number of remaining chars of other contributors of the touched article (in the case of removed content). In the following example, we only show the end result, that is after having run Module 5 of the cascade, for the two revisions. In our example, the `remaining char` value for the user-article pair `(2, 1290)` has gone through 3 states:

1. Revision 6738, Module 4: the *remaining char* value is initialized to 0, it is not computed as Module 5 has not run yet.
2. Revision 6738, Module 5: the value is updated to 9.
3. Revision 6739, Module 5: the value has changed because of an edit by user 7, and is now updated to 5.

User	Article	Revision	char add	char del	remaining char
2	1290	6738	9	0	5
7	1290	6739	6	4	6

One challenge is to compute all these data and measures incrementally so that a new revision will not need the re-computation of all the user or articles entries. This is why we have split our computations into modules. Also, by-products of the computations are very useful. For example, keeping the computed diffs allows programs such as History Flow [19] or Diffamation [4] to access the detailed information quickly — shorter transfer time due to the small size of the diffs and no extra computation compared to gathering the full text and computing diffs on their own. Note that the computed diffs are reversible: we keep the details of characters inserted as well as the characters removed. This allows applications to replay the diffs forward and backward, or to collected added and deleted words. Retrieving all the contributions for a specific contributor is also useful to compute the user's profile. Finally, keeping the sizes of the changes between revisions allows the visualization of a time-line of changes between revisions as in WikipediaViz [5].

5.4 Web Service

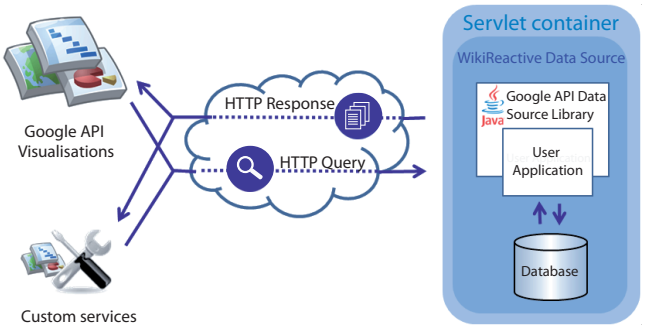

Figure 3: Google Visualization Data Source Library and our *WikiReactive* Implementation

To use *WikiReactive*, we currently offer a Web Service based on the Google Visualization Data Source[13] as depicted in Figure 3. This library provides a communication protocol and a simplified query language similar in syntax to SQL. A *data source* exposes

[13]http://code.google.com/p/google-visualization-java/

a URL, to which an external application, such as for visualization, can send data requests. In response, the data source returns properly formatted data that the visualization can use to render graphics on a web page. Using this web service, visualization widgets such as the ones described in WikipediaViz [5] can be implemented as JavaScript widgets and embedded into Wikipedia pages using the Google Visualization Widgets or any other visualization widget able to read the supported data formats.

The diagram in Figure 3 illustrates our adopted web server architecture. We run our *WikiReactive* Data Source as a servlet within an Apache Tomcat 5.5 container. When a visualization, such as a timeline, queries our data source, the servlet container handles the query and passes it to the Google API Data Source Java library for parsing. Our implementation code (the user application in Figure 3) returns a number of data tables to this library. Next, the library executes the query on the data tables. The response data is a two-dimensional table of values where every column is of a single type (e.g. text, number or date). The data table is then sent by the servlet container to the visualization that initiated the request.

The *request format* is an HTTP GET request with several parameters including an optional query string. The *response format* is a JSON formatted data table sent over HTTP because of the JavaScript based Google Visualization API. The library provides means for query optimization, e.g. when a client makes two requests, and the data has not changed between requests, data is not resent. For complicated or unsupported queries, data tables can be specified in a pre-compiled SQL query on the server side. The protocol supports the caching of data on the client side and sending a signal in the response if the data has not changed since the last request.

5.5 Initialization

WikiReactive requires an initialization phase for performance reasons: while it is in principle possible to download Wikipedia data through the MediaWiki API, it is very inefficient and would take months or even years for the largest languages. Fortunately, the Wikimedia Foundation provides regular dumps of the databases[14], encoded in XML and compressed. The initialization process consists in downloading two dump files for initializing the tables and processing them with initialization tools. After this phase, the polling programs can be run to maintain the continuous synchronization. Several dumps are provided for each Wikipedia database (i.e. for each language and service). Some contain only the latest versions of the pages, others contain all the versions. Some contain only the metadata; others also contain the full text. The largest dump contains all the versions of all the articles with full text.

The only constraint of the process comes from a limitation of Wikipedia: the recent changes are only available during one month. Therefore, if the polling mechanism of Wikipedia is started later than one month after the dump files have been created, the Wikipedia articles created in between will not be visible until they are changed again. Therefore, the initialization should be done as soon as possible after the dump files are made available.

5.6 Performance Statistics

Our *WikiReactive* implementation of the data collection and cascade modules is made of about 7000 lines of Java code. As for the user application code, it has 609 lines, only 78 lines of which describe the methods needed to provide direct access to the raw data tables.

We currently only run *WikiReactive* on the French version of Wikipedia for two reasons: the database is smaller and requires fewer resources than the English Wikipedia and, being in France, we can test our system easily. The French version of Wikipedia contains about 1 million articles.

[14]http://download.wikipedia.org/

Our server machine is a Quad-core Pentium at 3 GHz with 12 GB of RAM and 2TB of disk space, running Ubuntu Linux version 9.0. We use open source MySQL 5.0 database software.

In this section we describe relevant statistics regarding two distinct stages in launching *WikiReactive*; first, the initialization stage of the database where historic information from downloaded static files is taken into account; and second, the continuous update stage which synchronizes the data store with Wikipedia [15].

5.6.1 Initialization Stage

The first step in the initialization stage is to populate the Wikipedia state table (see Section 5.1) from the *current-meta-page* dump containing only the metadata of the last revision of each page. The French Wikipedia file of March 2010 is about 2 GB in size, about 5 times larger once uncompressed whereas the equivalent English file is 11 GB. It takes $\simeq$ 2.45 hours fill-up the state table for the French Wikipedia.

To bootstrap the rest of the pipeline for the whole Wikipedia, we need the largest dump file, *history-meta-page* — containing all the text of all the revisions — to compute all the diffs. This file is about 4.5 GB for the French Wikipedia and 32 GB for the English one, and is heavily compressed (about 10 times).

The next steps consist in getting the revision history of the articles from this large dump file and calculating the relevant aggregated information. The following statistics correspond to the run time of the modules cascade discussed in section 5.3:

1. **Diffs Computation** from the dump file takes $\simeq$ 2.5 days for more than 46.5 million article revisions;
2. **Per-User Contribution Computation** takes $\simeq$ 7 hours.
3. **Article Activity Computation** takes $\simeq$ 6.5 hours;
4. **User Contribution Computation** is included in the previous module and takes no time;
5. **Character Per Users Computation** is included in module 2 and takes no time.

The final initialization stage is to synchronize our local database with the live Wikipedia. We first query the MediaWiki API for the missing recent changes since the last revision time-stamp in our state table. It takes around 9 hours to reach the synchronized state, although the exact time depends on the lag between the latest time-stamp of current-meta-page file and the time this step is run (this can be up to one month). Then, we run our cascade modules to calculate the diffs and the aggregated information for the new revisions. It takes around 2.5 days to reach a synchronized state for all the remaining tables.

The figures described above should be roughly proportional to the size of Wikipedia and are bound by the computation speed for the diffs and by the network and the Wikipedia API for synchronizing with the live version of Wikipedia. Note also, that the content of the articles have to be retrieved from the API before computing the diffs, which is time consuming.

5.6.2 Continuous Update Stage

On average, we receive 15 article revisions per minute from the Wikipedia API, 21 if we count all Wikipedia domains (minimum is 1 and maximum is 65). We can process 2 revisions per second (cascading through all the different steps described in section 5.3). Modules run for-ever at this stage with sleep periods when there are no entries to process. For example, the sleep time of page collection process is 4 seconds and the sleep time of the diff computation process is 2 seconds.

[15]Note that many of the processes at these stages are run in parallel.

WikiReactive is now running continuously and has a very short lag with the French Wikipedia. It is therefore usable for supporting in real-time some of the needs expressed by Wikipedians.

Scaling *WikiReactive* to the English Wikipedia would be possible provided we have larger hard disks and a faster server. In principle, the initialization process could be parallelized but we have not designed it to support parallelization effectively. Therefore, it would take about 10 days to load the English Wikipedia, a long time but still short enough to avoid missing articles in the recent changes.

As for continuously updating from the English version of Wikipedia, we need to check if the current synchronization mechanism would scale. If not, we could use the Toolserver, which would provide higher throughput at the cost of higher administrative burden to get registered and authorized to access the database. Currently, WikiReactive can be installed and deployed without asking any authorization from Wikipedia.

6 EXAMPLES

In this section, we show examples of visualizations that have been provided using the *WikiReactive* infrastructure.

6.1 WikipediaViz

We have re-implemented the WikipediaViz visualizations [5] dashboard as a Wikipedia skin that every registered user can use. It is a service aimed at casual users that reveals important facts about the trustworthiness of articles.

Figure 4: Wikipedia with a WikipediaViz skin using *WikiReactive* showing the "Mythologie" article.

Figure 4 shows an example of an article editing profile, including the number of words of the article, the contribution pie chart, the timeline, the number of wikilinks and the number of words in the discussion.

The whole dashboard can be integrated as a new box in the left column of the classical Wikipedia interface by invoking a JavaScript widget in one user's skin using Monobook[16]. Each visualization is generated as html code through a PHP custom service and is embedded into an iframe in the skin. The JavaScript widget queries the PHP service by passing the visualization type (contribution, timeline or gauge) and the article name as parameters and obtains the visualization graphics as a result. The PHP service relies on data retrieved directly from the Wikipedia API (gauge), or by querying our *WikiReactive* Web service for aggregated information.

[16]http://en.wikipedia.org/wiki/Help:User_style

6.2 User Contribution Profile

WikiReactive provides several ways to help Wikipedians assess "at a glance" the profile of a user who appears as a new contributor in an article they monitor. The WikiTrust information (number of character remaining) is readily available, along with the total number of characters inserted, deleted and moved. Showing word frequency of users' contribution (e.g. as a tag cloud) can also help understanding their predominant style of contribution or topics of interest.

Figure 5: An example of a simple word cloud. Frequent but short contributions such as this may lead Wikipedians to spot a vandal. In this case, the user hid the content of a page.

The tag cloud in Figure 5 is based on the characters added by a user to an article over 21 edits. Coincidentally, the only text entered by this user was to hide the content of a page. This image clearly shows that this contributor is a vandal. To generate this type of graphic, we use the Google Visualization widget *Word Cloud*. This chart was easy to connect to our infrastructure as there are no additional plug-ins required in order the play the visualization.

Figure 6: A word cloud of one user's deleted words from 500 articles. Here, the user is a patrol bot.

For larger amounts of text, Google WordCloud is rather limited. We use Wordle [17] to generate the example in Figure 6. Here, the input data are the words deleted by a bot. Such clouds are currently not rendered on the fly but providing a dynamic service would not be difficult as the underlying data can be easily retrieved.

6.3 Overall User Contribution Over Time

For overview information on the activity of Wikipedia, useful for administrators of the Wikipedia Foundation trying to understand the activity over time or for sociologists studying the evolution of Wikipedia, *WikiReactive* is able to provide interesting overview visualizations such as the time series in Figure 7. In this example, the blue timeline denotes the total number of added characters, the red timeline denotes the deleted characters and the green timeline denotes the moved characters by all users over the lifetime of Wikipedia.fr. This interactive line chart is provided by the Google Visualization API and is easily plugged in to our web service infrastructure. The chart sends requests to the data source and renders the data within the browser using Flash. The refresh time of the chart can be specified such that new user edits are taken into account in a timely fashion.

With this tool, Wikipedians can monitor the overall user contribution over the lifespan of Wikipedia in real-time. It can highlight unusual user activities and describe the evolution pattern of articles over time. Overall, contributions to Wikipedia continue to grow as

[17]http://www.wordle.net

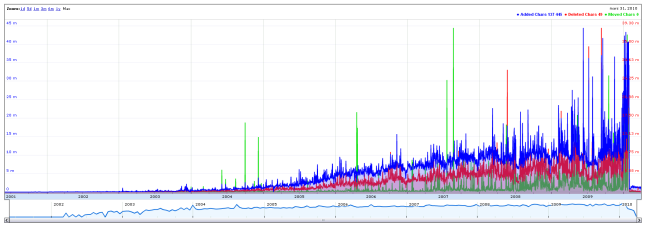

Figure 7: Timeline series showing added (in blue), deleted (in red) and moved (in green) characters for all articles over the lifespan of Wikipedia.fr.

users add more characters than they would delete. However, there are occasional peaks where large amounts of text are deleted (e.g. 39 million chars on 11-09-2009); or moved (e.g. 998 million chars on 23-08-2007).

7 Conclusion

In this article, we have described the *WikiReactive* infrastructure we have designed to compute aggregated information about Wikipedia in real-time. The information computed is aimed at helping the Wikipedia community improve its process and raise the quality and coverage of Wikipedia.

WikiReactive collects and aggregates data from Wikipedia and provides it live through a Web service. This is different from existing systems that either directly use the data available from Wikipedia but cannot aggregate it in real-time, or aggregate data from Wikipedia using the tool server but only present it on a Web page using a predefined representation. *WikiReactive* is meant to provide data for Visual Analytics applications on Wikipedia.

Although the focus of this research was the development of the *WikiReactive* infrastructure, we have also presented several useful visualizations provided by our group or others that benefit from the data provided by *WikiReactive*. Much more can be done in terms of visualizations and uses and we hope to see them soon available for Wikipedians.

As an infrastructure, *WikiReactive* can also be extended to compute more sophisticated aggregated data and we will work on adding new services. Important ones include text-mining computations to help Wikipedians find relevant articles to link when entering a new article or to adequately associate articles to categories, categories to articles and clean-up the Wikipedia categories that are sometimes complex. We also intend to carry out a longitudinal study to assess the usefulness of our visualizations and the underlying infrastructure. More precisely, we would like to investigate how dashboard-type visualizations affect the time Wikipedians spend on each of their main tasks.

The problem of computing aggregated information on large databases maintained by a large group of people with no hierarchical relations is not specific to Wikipedia — although Wikipedia is certainly the largest of these communities yet. Other domains such as Biology or Chemistry develop large databases that started as centrally controlled by curators and are turning now to a more open model of curation to support their growth. These databases will also benefit from an infrastructure similar to *WikiReactive* to propose new visualizations for activity awareness.

WikiReactive is available as a free service and the infrastructure as an open-source software at `http://www.aviz.fr/wikireactive`.

Acknowledgements

The authors wish to thank Tomer Moscovich, Damien Schroeder, Danyel Fisher, Nathalie Henry-Riche, Johan Euphrosine and Xiujun Li, for their help and fruitful discussions on the project. Thanks to the Wikipedians who participated to our sessions and helped us during this work. This work was supported by Microsoft Research.

References

[1] B. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *WWW'07: Proc. of the 16th international conference on World Wide Web*, pages 261–270, 2007.

[2] J. Blumenstock. Size matters: word count as a measure of quality on wikipedia. In *WWW'08: Proc. of the international conference on World Wide Web*, pages 1095–1096, 2008.

[3] K. Chatterjee, L. de Alfaro, and I. Pye. Robust content-driven reputation. In *AISec '08: Proc. of the 1st ACM workshop on Workshop on AISec*, pages 33–42. ACM, 2008.

[4] F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete. Using text animated transitions to support navigation in document histories. In *CHI'10: Proc. of the SIGCHI Conference on Human factors in computing systems*, pages 683–692. ACM, Apr. 2010.

[5] F. Chevalier, S. Huot, and J.-D. Fekete. WikipediaViz: Conveying article quality for casual wikipedia readers. In *PacificVis'10: IEEE Pacific Visualization Symposium*, pages 49–56, 2010.

[6] D. Cubranic and G. C. Murphy. Hipikat: Recommending pertinent software development artifacts. In *Proc. of the 2003 Int. Conf. on Software Engineering*, pages 408–418, 2003.

[7] P. Dondio, S. Barrett, S. Weber, and J. Seigneur. Extracting trust from domain analysis: A case study on the wikipedia project. *Autonomic and Trusted Computing*, pages 362–373, 2006.

[8] G. Druck, G. Miklau, and A. McCallum. Learning to predict the quality of contributions to wikipedia. In *WIKIAI 08*, pages 7–12, 2008.

[9] D. M. German, A. Hindle, and N. Jordan. Visualizing the evolution of software using softchange. In *SEKE'04: Proc. of the Int. Conf. on Engineering and Knowledge Engineering*, pages 336–341, 2004.

[10] P. Heckel. A technique for isolating differences between files. *Commun. ACM*, 21(4):264–268, 1978.

[11] A. Kittur, B. Suh, and E. Chi. Can you ever trust a wiki? impacting perceived trustworthiness in Wikipedia. In *CSCW '08: Proc. of the Conf. on Comp. supported coop. work*, pages 477–480, 2008.

[12] M. Kramer, A. Gregorowicz, and B. Iyer. Wiki trust metrics based on phrasal analysis. In *WikiSym'08*, 2008.

[13] A. Lih. Wikipedia as participatory journalism: Reliable sources? metrics for evaluating collaborative media as a news resource. In *5th International Symposium on Online Journalism*, 2004.

[14] P. Pirolli, E. Wollny, and B. Suh. So you know you're getting the best possible information: a tool that increases Wikipedia credibility. In *CHI'09: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 1505–1508, 2009.

[15] J. Preece and B. Shneiderman. The Reader-to-Leader framework: Motivating technology-mediated social participation. *AIS Transactions on Human-Computer Interaction*, 1(1):13–32, 2009.

[16] N. Riche, B. Lee, and F. Chevalier. iChase: Supporting exploration and awareness of editing activities on wikipedia. In *AVI'10: Proc. of the Int. Conf. on Advanced Visual Interfaces*, pages 59–66, 2010.

[17] J. Schiefer, J.-J. Jeng, and R. M. Bruckner. Managing continuous data integration flows. In *DSE'03*, 2003.

[18] B. Suh, E. H. Chi, A. Kittur, and B. A. Pendleton. Lifting the veil: improving accountability and social transparency in Wikipedia with WikiDashboard. In *CHI'08: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 1037–1040. ACM, 2008.

[19] F. B. Viégas, M. Wattenberg, and D. Kushal. Studying cooperation and conflict between authors with History Flow visualizations. In *CHI'04: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 575–582. ACM, 2004.

[20] S.-L. Voinea. *Software Evolution Visualization*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[21] M. Wattenberg, F. Viégas, and K. Hollenbach. Visualizing activity on Wikipedia with Chromograms. In *INTERACT'07*, pages 272–287, 2007.

[22] D. M. Wilkinson and B. A. Huberman. Cooperation and quality in Wikipedia. In *WikiSym'07: Proc. of the 2007 international symposium on Wikis*, pages 157–164, 2007.

[23] T. Wöhner and R. Peters. Assessing the quality of wikipedia articles with lifecycle based metrics. In *WikiSym'09: Proc. of the Int. Symp. on Wikis*. ACM, 2009.