Addressing the Cold Start Problem in the Wikipedia Recommender System through Content-Based Filtering

Mihai Mihăilă

Kongens Lyngby 2011

IMM-MSC-2011-75

Technical University of Denmark

Informatics and Mathematical Modelling

Building 321, DK-2800 Kongens Lyngby, Denmark Phone +45 45253351, Fax +45 45882673 reception@imm.dtu.dk

www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Abstract

Wikipedia is a free online encyclopaedia that can be edited by anyone. Despite its large number of contributors and articles, the qualification of the authors, as well as the quality of the contribution they generate, cannot be easily assessed.

Wikipedia Recommender System is a collaborative filtering system used to determine the quality of Wikipedia articles based on user ratings. Being a collaborative filtering system, WRS is affected by the cold start problem. The cold start problem is a phenomenon that occurs when there is not enough data for the system to function correctly. New users of WRS do not receive article ratings until they start interacting with the system and have a trust profile. The current project addresses this problem by using WikiTrust article rating.

WikiTrust is a system that calculates the rating of an article by determining and computing the trust value of its words. In this system, the words' trust level is proportional to their authors trust. Authors gain trust by making changes that last over multiple reviews.

The goal is to use the WikiTrust article reputation for calculating the WRS trust level in those cases when WRS does not have enough information to determine a good trust estimation of the article quality.

While the main purpose is to address the cold start problem, incorporating WikiTrust reputation into WRS trust calculation could potentially increase WRS's overall trust level accuracy. The current project will investigate the advantages and disadvantages of integrating the two systems, WRS and WikiTrust, and will attempt to determine the best formula to use the latter for improving the current system performance.

Preface

This thesis was prepared at the Department of Informatics and Mathematical Modelling, within the Technical University of Denmark in partial fulfilment of the requirements for acquiring the M.Sc. degree in engineering.

The project was completed in the period from March 7th, 2011 to September 23th, 2011 under the supervision of Associate Professor Christian Damsgaard Jensen.

The project design was presented during the poster session of International Conference on Trust Management 5th IFIP WG 11.11.

One of the results of this project, the WRS Google Chrome Extension, has been published in the Chrome Web Store and is currently available for free download[1] and usage.

Lyngby, September 2011

Mihai Mihăilă

s091368

[1] Wikipedia Recommender System - Chrome Web Store
https://chrome.google.com/webstore/detail/dlbpjdiahnhhokdbanadnhgjfoiojdmb

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

1.1. Introduction

Wikipedia is a free, web-based and collaborative encyclopaedia. It was founded in 2001 and since then over 3.7 million[2] articles have been created. It currently has over 90,000 active contributors. Given its size and the variety of its topics it has become one of the most visited websites on the Internet. The popularity of Wikipedia heavily relies on its openness, as virtually anyone can change an article or create a new one. Although there are some ground rules and special requirements that every editor should follow, a lot of controversy exists around Wikipedia.

Critics often underline the fact that the authors of Wikipedia are regular internet users with no proven qualification to write a reference or scientific paper in the first place. Moreover, as it can be edited by anyone, it is often the case that certain articles are edited in such a way as to reflect one’s subjective beliefs or they are inspired by potentially untrustworthy source like TV, radio, personal blogs, etc. This situation

[2] More precisely 3,709,225 articles at 2011-08-14

leads to a lack of credibility of the article that cannot be quoted or used for any academic or scientific purpose.

Another problem occurs along with the editing process. Typically, groups of editors built around articles tend to preserve their contributions and resent outsiders with different opinions. In the latest annual Wikipedia meeting, Jimmy Wales, one of the co-founders of Wikipedia, acknowledged this problem as affecting the number of editors and especially the number of new editors and consequently he announced measures to simplify the editing procedures. Meanwhile, the issue is still present and editing Wikipedia pages is governed by its 1,544 volunteers, possibly biased, administrators.

Larry Sanger, the other co-founder of Wikipedia, stated in an article back in 2004 that the source of all this problems was the lack of respect for expertise (Sanger, 2004). Currently, Wikipedia does not have any mechanism for distinguishing between common and expert users; hence the bad reputation it has in terms of credibility and information accuracy.

By promoting the openness idea, Wikipedia has encouraged people to contribute, assuming that over time articles will reach a mature state. At that point they can be moved into a *featured*[3] state, where only minor changes are accepted. However, there are only few featured articles on Wikipedia; the ratio between normal and featured articles is about one to 1100.

While the credibility of Wikipedia articles is put into question, its widespread use is a reality that cannot be ignored. People are using it for various purposes even if they are aware of its problems. According to alexa.com[4], Wikipedia is the 7th most visited website on the Internet (August 2011).

In this context, there has been an acute need for assessing the quality of articles in the Wikipedia. Various studies have shown that some articles on Wikipedia have the same level of correctness as an academic article on the same topic (Giles, 2005). The main proof of this is in the phrasing and the flow of the article, which is in favour of the academic one. At the same time, articles on controversial topics or recent events may have a very low quality and sometimes contain wrong, misleading information. It is therefore necessary to highlight the differences between these two types of articles for increasing the credibility of Wikipedia and the accuracy of the information its visitors use.

[3] An article reviewed by the community and promoted as having a very good quality. A featured article is marked with small bronze star icon on the top right of the article's page. There are 3350 featured articles on Wikipedia (August 2011).
[4] http://www.alexa.com is an internet ranking system

1.2. Wikipedia Recommender System

The Wikipedia Recommender System (WRS) is a collaborative filtering system which can be used alongside Wikipedia to determine the quality of an article. It utilises the user's article ratings in order to detect similarities between users and provide a personalized, subjective rating for an article.

The system creates a user trust profile and records the user's ratings along with other user ratings on visited articles. To do so, trust metrics are used. By identifying users with similar ratings the system can make predictions for other articles as well, better than traditional filtering systems. Such a system typically averages the ratings across a whole community of users and makes its recommendations based on it. WRS uses trust metrics to determine the users with a similar trust profile and then calculates an overall rating, based only on their ratings. Basically, it selects similar users from a particular community when calculating a rating, rather than the whole community.

In the past decade, this approach has retained a lot of attention in academia as well as industry. Several systems like *ebay.com*[5] or *slashdot.org*[6] have been using the same or similar principles for building reputations for their community members.

Recently, bigger names in the industry seem to have made the first steps into adopting the same principles even when it comes to search algorithms:

In May 2011 *Bing*[7] acknowledged the fact that 90% of their users (which currently represents around 15% of the market share in internet search) seek advice from friends or family in their decision process. As a consequence, they decided to incorporate *Facebook*[8]'s *Like*[9]-ed pages in their search, giving them priority over other similar pages[10]. As friends and family members typically have similar preferences with a user, we see this move as an attempt to integrate trust and personal preferences into the traditional search results.

In August 2011, following the same principle as Bing, *Google*[11] announced that its search results will integrate and prioritize the posts in Google Plus[12].

---

[5] http://www.ebay.com/ - An online auction and shopping website
[6] http://slashdot.org/ - an online technology-related news website
[7] http://www.bing.com – the 3rd biggest search engine on the Internet, August 2011
[8] http://www.facebook.com – the largest social network on the Internet, August 2011
[9] Facebook Like is user's ability to express his recommendation for an internet resource or other internal items (comments, pictures etc.)
[10] (bing.com, 2011)
[11] http://www.google.com – the biggest search engine on the Internet, August 2011

These widely used systems gather information from social networks, on the premise that those networks are built on top of similarities between its users. Using social data is therefore a clear indicator of the importance that similar preferences have, when it comes to collaborative filtering.

However, WRS does not rely on existing social networks. Instead, it determines users with similar trust profiles based on the ratings they give to Wikipedia pages. User ratings are stored in special pages of Wikipedia and are publicly available. Personal trust profiles are stored on user’s local machine, while for providing a feedback mechanism it uses a proxy filter for injecting an UI into Wikipedia pages.

1.3. Objectives

Any collaborative filtering system is affected by a phenomenon called *cold start problem*[13].

A collaborative filtering system relies on the network’s knowledge for assigning a rating for an entity for which the user does not have any knowledge about yet. The strength of these systems grows as the network contributing to the rating is larger, therefore theoretically more accurate. The problem emerges when the network is small or it does not exist at all. In these cases, the calculated rating might be inaccurate or cannot be calculated at all. In this situation, the system meets a typical cold start problem.

There are several ways of dealing with the cold start problem, among which inferring a rating from an extended network of indirect friends or connecting to already existing networks are the most common. However, these approaches either require user’s intervention (for connecting to an existing network) or rely on the fact that, by following a network connection, some knowledge will ultimately be available about the current entity. It seems that the scenario where the user has no network or is part of an isolated network, that has no knowledge about the current entity, is not covered by any of them.

As a consequence, we turn our attention towards an external system that can always provide a rating for an entity, in this case, a Wikipedia article.

[12] Google Plus is Google’s social network
[13] (Schein, et al., 2002)

WikiTrust[14] is a filtering system developed at University of California, in Santa Cruz, United States, which provides a rating for any Wikipedia article, based solely on its content. WikiTrust article trust level is determined by the trust level of its contributions. A contribution gains trust if it is created by a high reputation author and lasts over multiple reviews. An author gains trust by creating contributions that last over time (therefore, having an increased trust level).

WikiTrust can be used by itself as a Firefox add-on[15], which informs the user about the article level of trust by using colour coding (words' background ranges from white – high trust, towards dark orange – low trust).

WikiTrust also provides a rich API for retrieving article trust level, author trust level and other related functions.

The current approach for solving the cold start problem in WRS consists in integrating the WikiTrust article rating for those cases where no article rating is available.

There are several challenges in achieving this goal, among which:

- Dealing with the *general rating* WikiTrust provides opposite to *category-specific rating* WRS uses.
- Investigating the WikiTrust ratings in terms of range, compared with the WRS ratings.

The goal of the thesis is to address these challenges and incorporate WikiTrust ratings into WRS for overcoming the cold start problem. However, after analysing the system structure and starting the development process, various problems have been identified, that were blocking the development, as described in the next chapter, *State of the Art*. Therefore, in order to achieve the main goal of the thesis, we first have to modify the existing functionality, so that WRS can be used with no restrictions or limitations by any user. Secondly, we aim to maintain and improve (where possible) the current user interaction, the installation complexity and the distribution method.

1.4. Structure

The current thesis is structured as follows:

[14] http://www.wikitrust.net/
[15] https://addons.mozilla.org/en-US/firefox/addon/wikitrust/

Introduction describes the WRS project along with the objectives of the thesis.

State of the Art gives an overview of the current state of the project while highlighting the current issues that are to be fixed.

Analysis presents a strategy for fixing the cold start problem and the current issues introduced in *State of the Art*. WikiTrust system is viewed as an argument for integrating it with WRS.

Design discusses the proposed solutions for achieving the goal of the thesis (fix the cold start problem in WRS) and the design changes introduced in *Analysis*.

Implementation brings forward the specific ways in which the proposed architecture change was achieved. The most important system components are discussed in detail, while presenting the differences and similarities with the previous implementation where necessary.

Evaluation shows the state of the project after the proposed changes have been applied by highlighting individual achievements. It also contains the results of multiple sets of tests that have been performed in order to assess the ratings of WikiTrust as a working part of WRS.

Conclusion summarises the results achieved in this thesis and proposes various areas for future work and research.

Appendix contains the source code and other resources used in the project.

1.5. Definition of Terms

In this section a set of terms will be defined as they are used in the current thesis.

Trust: One’s personal and contextual opinion about a specific subject.

Rating: The value WikiTrust returns as a quality indicator for an article or the quality indicator a user assigned in the WRS system to an article.

Reputation: A group’s contextual opinion about some specific subject.

WRS: Wikipedia Recommender System is a recommender system for Wikipedia that calculates ratings based on similarities between users. The similarities are based on the ratings the users of the system assign to articles.

WikiTrust: An online content-based filtering system that can provide a rating for any Wikipedia article[16].

Trust value: In WRS, trust is measured on a $[-1,1]$ scale (-1 means complete distrust and 1 means complete trust) and represents the degree in which a trustor trusts a trustee.

Trustor: The current user of WRS, who reads a Wikipedia article and is interested in receiving a rating for it.

Trustee: Any user of the WRS user community different from the trustor.

Web of Trust: User's network of users for which he has a trust value.

Trust profile: The WRS representation of user's trust, including all his previous interactions with other users and the data which contributes to a trust value.

[16] The supported Wikipedias are English, French, German, or Polish editions.

Chapter 2

2. State of the Art

In this chapter we look at the current system, which is the working result of (Korsgaard, 2007), (Lefevre, 2009) and (Pilkauskas, 2010) (in chronological order of their contributions).

In *Trust Model* and *Classification* we present the theoretical foundations of the WRS.

In *Current Architecture* we look at different key parts of the system and the way they work together.

2.1. Trust Model

WRS tries to mimic the human behaviour in relation with trust. Individuals experience trust in most of their social interaction which helps them establish relationships. There are a series of factors involved in modelling trust in its representation used by WRS.

Initial Trust

WRS's trust model is based on the model proposed by Stephen Marsh (Marsh, 1994). In this model, trust is represented on a $[-1,1]$ range, -1 meaning complete distrust, and 1 meaning full trust. The trust value for a new user is initialized with 0.0, as no information is known about him. This initial value means the user is neither trusted, nor distrusted.

Trust Dynamics

Over time, the trust value is modified by the user interactions. The model adopted by WRS takes into account the order of the interactions and their age. The interactions that occurred in the past count less than the ones that have occurred recently. The interactions that are less than a month old count 100%, those between one month and six months count 50% and the ones between six months and one year count 25%. Interactions older than one year are ignored.

An interaction is worth an absolute value of $\frac{1}{10}$ which gives 20 steps between -1.0 and 1.0, which are the boundaries of the trust values.

Trust Evolution

The core of the WRS trust model is represented by the trust evolution function. Its formula is based on a superellipse:

$$\left|\frac{x}{a}\right|^n + \left|\frac{y}{b}\right|^n = 1$$

Where:

- *x* is the sum of interaction
- *y* is the calculated trust value

As the function is based on the model proposed by Stephen Marsh (Marsh, 1994), *a* and *b* parameters, which give the radius of the superellipse, are equal with 1, therefore a = 1 and b = 1. The *n* parameter which gives the curvature of the curve starts at 1. This parameter will be updated based on whether the user is cautious or optimistic. n < 1.0 gives a cautious curve, while n > 1.0 gives an optimistic curve.

Four functions are used to describe the possible scenarios in WRS:

- An optimistic curve in trust:

$$|x-1|^n + |y|^n = 1, where\ x\ \in [0.0, 1.0]\ and\ y \in [0.0, 1.0]$$

- A cautious curve in trust:

$$|x|^n + |y-1|^n = 1, where\ x\ \in [0.0, 1.0]\ and\ y \in [0.0, 1.0]$$

- An optimistic curve in distrust:

$$|x|^n + |y+1|^n = 1, where\ x\ \in [-1.0, 0.0]\ and\ y \in [-1.0, 0.0]$$

- A cautious curve in distrust:

$$|x+1|^n + |y|^n = 1, where\ x\ \in [-1.0, 0.0]\ and\ y \in [-1.0, 0.0]$$

These equations, when n = 2, produce the curves in *Figure 1: Trust evolution function*.

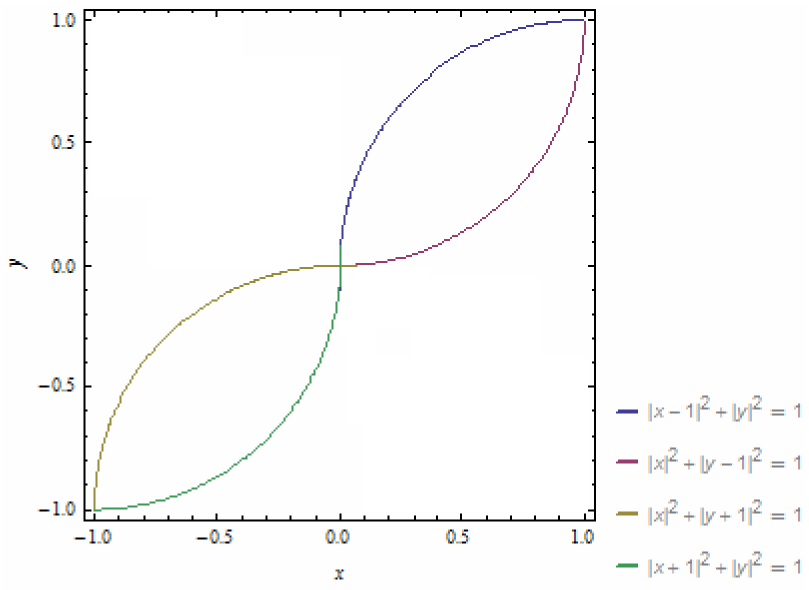


Figure 1: Trust evolution function

2.2. Classification

WRS's goal is to create a web of trust for its users in order to be able to predict rating values for unknown articles to the trustor, but rated by the trustee.

Trustor's trust in a trustee is contextual, associated with a category. When rating an article, the trustor places it in one of the available categories. By assigning categories to ratings the system tries to mimic the real life experience where people have contextual trust in others.

For example, a person A can trust another person B when it comes to movies preferences, but he might not agree with B's food preferences, if only one of them is vegetarian.

Figure 2: Trust relationship between user A and B illustrates a similar example, where user A has a high trust in user B in *Computers* category as they have given the same rating to the *Microsoft* article. On the other hand, A's trust in B is low in *Sports* category as B's rating is different than his.

Notice that WRS, instead of having prior information about the relationship between A and B, looks at ratings rated by both A and B in order to understand how close they are in terms of preferences. By doing so, it eliminates the need of having to preconfigure the system.

Obviously, there are far more complex factors in real life that contribute to building trust. However, the adoption of rating categories is the first step into achieving contextual trust.

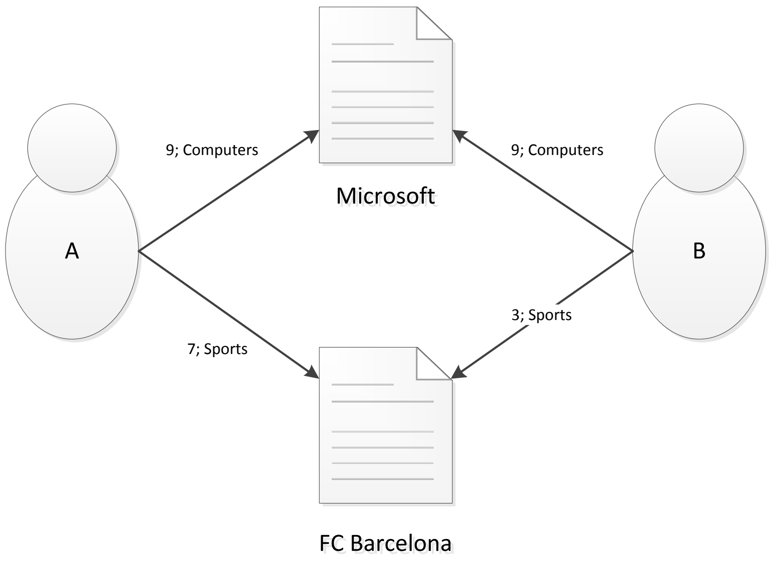


Figure 2: Trust relationship between user A and B

In this same figure, WRS trust seems to be bidirectional. However, given the way trust is calculated, it is in fact unidirectional. WRS updates the trust values only if it finds a prior user rating while visiting a page. Therefore, users will have different trust values in each other.

WRS uses the same top 15 categories as Open Directory Project[17]. They were found by Pilkauskas (Pilkauskas, 2010) to be simple enough for the average user while best covering the wide range of Wikipedia articles.

2.3. Current Architecture

WRS is currently implemented as a plugin for the Scone Proxy. It uses Wikipedia user pages as a central repository for user ratings and it computes and stores the user's trust profile on the local machine. An overview of the system can be observed in *Figure 3: Scone Proxy WRS Architecture*.

[17] http://www.dmoz.org/about.html

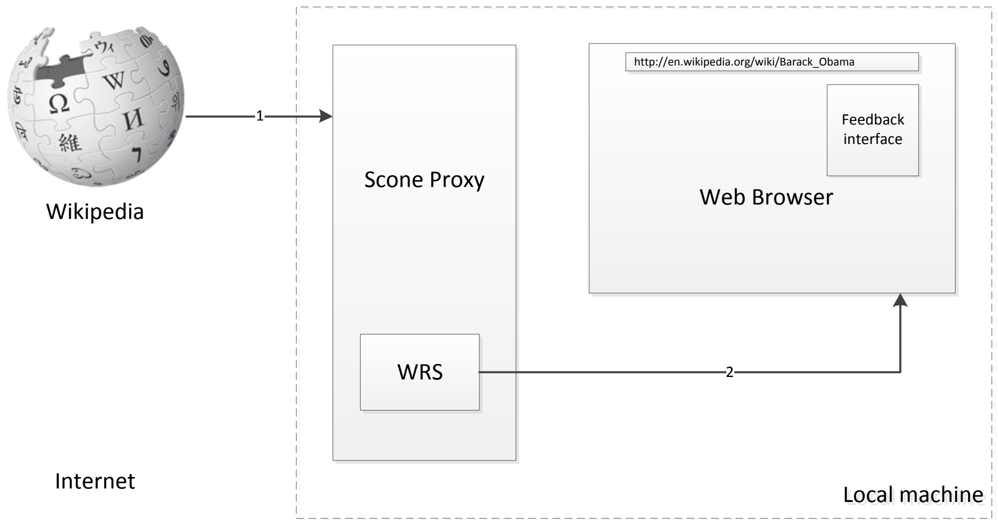


Figure 3: Scone Proxy WRS Architecture

The Scone Proxy

The Scone Proxy[18] is a Java Framework designed to allow the development of web enhancements for research and educational purposes. It acts as an intermediary between webpages and the web browsers, capturing the browsing data, allowing its manipulation and then forwarding it to the web browsers.

Being a Scone Proxy plugin, WRS exposes its functionality to a variety of browsers. WRS intercepts Wikipedia pages and manipulates them in order to display a page rating and a feedback mechanism.

The Scone Proxy is based on IBM's Web Intermediaries (WBI) technologies. The latest version of WBI was released in June, 1999.

As mentioned by Lefevre (Lefevre, 2009), Scone Proxy had not seen a good adoption by the community. The project's website[19] is only mentioning a limited number or prototypes based on Scone.

[18] http://www.scone.de/index.html
[19] http://www.scone.de/examples.html

Scone Proxy's latest version was released in February, 2009. The fact that the underlying technology (WBI) has not been under development for over a decade, along with the fact that Scone's tested platforms[20] and browsers (Internet Explorer 6, Firefox 1.x, Opera 7.x) are far from recent. So, while still working, the Scone Proxy is a thing of the past.

Wikipedia as a Central Repository

In its early stages WRS used the article's pages to store ratings as wiki comments (therefore, not visible to the users reading the article in order to avoid having to maintain another online system to serve as a repository. This approach was quickly dropped after Wikipedia community complained about WRS generating unneeded content for all users, even the ones that were not using WRS.

The next step was to look at other locations on Wikipedia that could store the user ratings, without interfering with the actual article content. This location was identified in user pages.

Every Wikipedia user has access to a user page which can be used for drafts, personal notes as well as any other content compatible with Wikipedia purpose. The user pages act as any other Wikipedia page in terms of contributions, meaning that anybody can edit such a page.

As the WRS goal is to improve Wikipedia's functionality, WRS uses these user pages as a central repository for ratings. To do so, a user named *Recommendations* was created and all the ratings were kept in this user's subpages. Therefore, as the user page is located at `http://en.wikipedia.org/wiki/User:Recommendations`, the ratings for an article like `http://en.wikipedia.org/wiki/Winter` are kept at `http://en.wikipedia.org/wiki/User:Recommendations/Winter`.

However, as the Wikipedia policies tightened over time to prevent automatic editing tool (which had a potential of generating unneeded, wrong or malicious content) as well as bad contributions to wiki pages, this approach (of storing ratings in the user pages) faced some challenges as well.

In May 2011, the use of WRS generated controversy among Wikipedia administrators which blocked the accounts creating the ratings and deleted the user pages containing the ratings. Such a situation was caused by two user accounts who rated

[20] http://www.scone.de/download.html

two different articles, hence generating the automatic creation and subsequent edition of two use subpages.

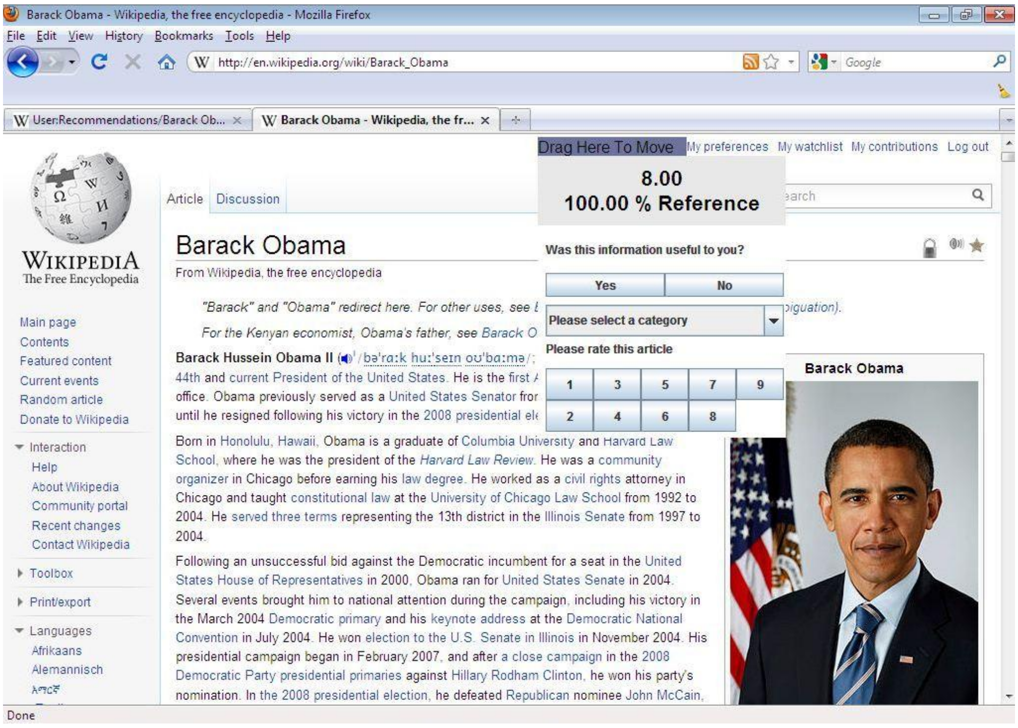
After explaining to the administrators the purpose of WRS and the way it worked, the user accounts were unblocked, but several restrictions were imposed, which conflicted with the way WRS worked.

The accounts that initially used WRS for browsing and rating pages were consequently no longer able to use WRS. This was a clear indicator that the current strategy for storing ratings on Wikipedia had to change.

Feedback Interface

The system is interacting with the user through its Feedback Interface illustrated in *Figure 4: WRS Feedback interface*. This visual element which in this example has a dimension of 250*350 pixels is always added to the Wikipedia articles and cannot be closed or hidden. The Feedback Interface can be moved around, but, by doing so, it fails to update its visual content for several seconds or until the screen is updated.

Another problem of the current Feedback Interface is the placement of the rating buttons. The buttons used for rating are placed on two rows, with a top – down, left – right arrangement. This placement is confusing and contradicts basic UI rules as the user has to follow a zigzag path in order to find a specific button.



8.00
100.00 % Reference

Was this information useful to you?

Please rate this article

Barack Obama

From Wikipedia, the free encyclopedia

"Barack" and "Obama" redirect here. For other uses, see I ›iguation).

For the Kenyan economist, Obama's father, see Barack O

Barack Obama

Barack Hussein Obama II /bəˈrɑːk huːˈseɪn oʊˈbɑːmə/;
44th and current President of the United States. He is the first /
office. Obama previously served as a United States Senator fror
until he resigned following his victory in the 2008 presidential ele

Born in Honolulu, Hawaii, Obama is a graduate of Columbia University and Harvard Law School, where he was the president of the *Harvard Law Review*. He was a community organizer in Chicago before earning his law degree. He worked as a civil rights attorney in Chicago and taught constitutional law at the University of Chicago Law School from 1992 to 2004. He served three terms representing the 13th district in the Illinois Senate from 1997 to 2004.

Following an unsuccessful bid against the Democratic incumbent for a seat in the United States House of Representatives in 2000, Obama ran for United States Senate in 2004. Several events brought him to national attention during the campaign, including his victory in the March 2004 Democratic primary and his keynote address at the Democratic National Convention in July 2004. He won election to the U.S. Senate in Illinois in November 2004. His presidential campaign began in February 2007, and after a close campaign in the 2008 Democratic Party presidential primaries against Hillary Rodham Clinton, he won his party's nomination. In the 2008 presidential election, he defeated Republican nominee John McCain,

Figure 4: WRS Feedback interface

2.4. WRS Evolution

The first version of WRS was developed by Thomas Rune Korsgaard (Korsgaard, 2007). It used article's page to save user ratings.

The second version of WRS was developed by Thomas Lefevre (Lefevre, 2009). It introduced categories for the ratings as well as storing the ratings in user's page instead of article's page.

The third version of WRS was developed by Povilas Pilkauskas (Pilkauskas, 2010). It changed the number of the rating categories for the system to allow a better categorization scheme.

2.5. Summary

In this chapter we have looked at the current implementation of WRS and have highlighted some of its weaknesses. While analysing the system, we have noticed WRS is not functioning anymore because some changes in Wikipedia policies. These findings will serve as an argument for changing the system’s architecture as a first step towards achieving the goal of the thesis.

Chapter 3

3. Analysis

Our approach to solving the cold start problem in WRS is to use WikiTrust whenever there are no user ratings available. In this chapter we will look at WikiTrust system and discuss the specific ways it calculates trust as an argument for using it within WRS. The suggested solution does not cover all the aspects of the cold start problem especially if we consider cold start the state when the system is not relying on enough data to calculate an accurate rating. Our resolution is however a starting point upon which several other techniques can be built for a better performance of the whole system.

Due to the problems we found in *State of the Art*, we propose a series of architecture changes for WRS. These changes are presented in *Necessary System Architecture Changes*.

3.1. WikiTrust

As described by its authors, WikiTrust is a content-driven reputation system for Wikipedia authors. In this system, users gain reputation when their contributions are preserved over multiple reviews by other users, and they lose reputation when their contributions are changed, deleted or reverted. The system differentiates between various ways of preserving or removing one's changes, and updates the reputation accordingly.

WikiTrust Reputation

An interesting side of WikiTrust is its implicit trust aspect of the reputation, which makes it a good candidate for integrating it with our WRS system. WikiTrust calculates reputation based on *text life* and *edit life*, which are determined by human users[21]. Author B, preserving the previous changes of author A in his revisions, expresses his trust in author A's contribution. Author C, removing the changes of author A in his revisions, expresses his distrust in author A's contribution. Ultimately, WikiTrust reputation value for an author is the reputation within the group of authors with whom he collaborated on Wikipedia pages.

WikiTrust computes the trust value for each individual word (as part of a revision), which is based on its lifespan and on its author's reputation. As a consequence, any Wikipedia article has a rating in WikiTrust based on its containing words. Therefore a page rating is based on the trust values of all authors that contributed to the text of the page.

[21] Even if automatic software can make changes to Wikipedia as well

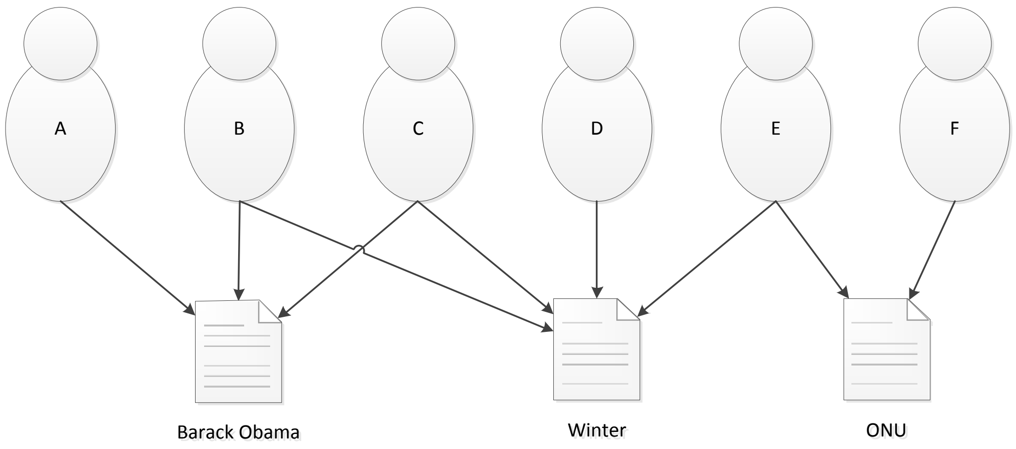


Figure 5: WikiTrust rating example

In *Figure 5: WikiTrust rating example* the rating for the article *ONU* is influenced by all the authors illustrated (A, B, C, D, E and F) even if only Author E and Author F have directly edited it.

Notice that:

- Authors B, C and D have potentially contributed to the reputation of Author E by contributing to the article *Winter*.
- Authors A potentially have contributed to the reputation of Author A and C by contributing to a common article, namely *Barack Obama*.

At the same time, we acknowledge the fact that this network of authors might not contain the user reading an article and interested in a rating. Therefore, integrating WikiTrust in WRS means automatically assigning the reputation of an article as perceived by the network of its authors and all the other authors contributing to their trust level. We believe this trust value is relevant for the quality of the article, therefore integrating WikiTrust into WRS can be perceived as connecting the user to a general rating generated by members of the network, most likely to have a meaningful opinion. This approach is expected to be better than other techniques for solving the cold start problem by connecting to a random network user (Victor, et al., 2008).

Prior to integrating WikiTrust into WRS, the Firefox WikiTrust Add-on has been tested in order to assess its correctness. The following has been observed:

- Featured articles have a generally high trust illustrated by a white background for most of its words.
- Poor quality articles[22] generally have a predominant orange background which suggests low trust especially in those sections concerning recent events.
- Introducing a small change into an article using a new Wikipedia account is illustrated as low trust by dark orange background.

Given the results presented by the authors of WikiTrust and the working Mozilla Firefox extension, as well as the manual testing performed, integrating WikiTrust into WRS is a motivated choice.

Using WikiTrust in WRS

WikiTrust will be used on top of the existing WRS implementation, in those cases where the system does not have enough data to calculate a rating. The workflow of the new system is illustrated in *Figure 6: Fixing cold start using WikiTrust*.

[22] Our criteria for finding poor quality articles will be detailed in *Evaluation.*

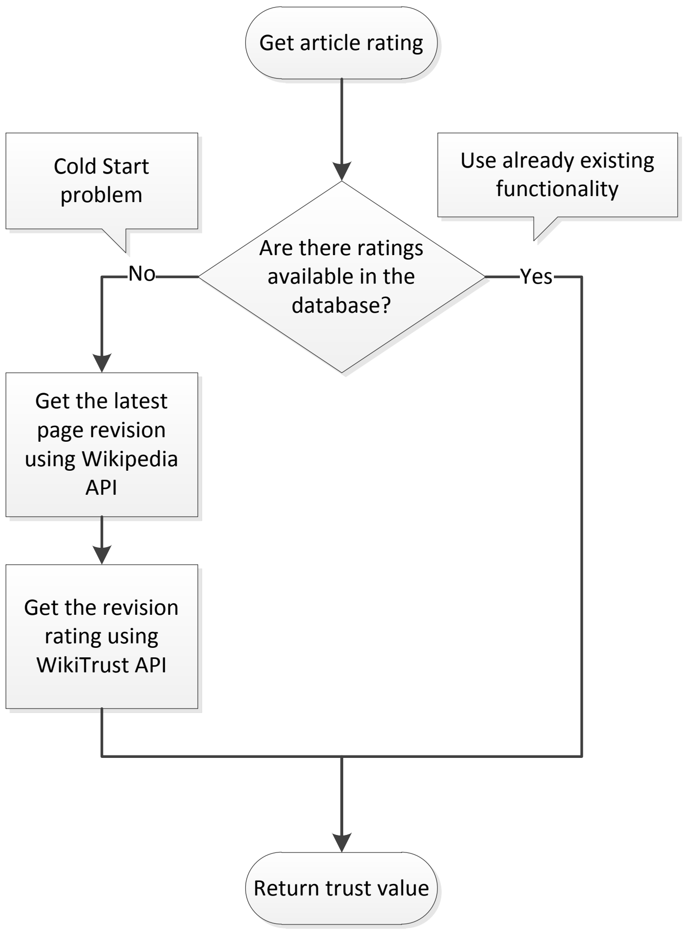

Get article rating
Cold Start problem
Use already existing functionality
Are there ratings available in the database?
No
Yes
Get the latest page revision using Wikipedia API
Get the revision rating using WikiTrust API
Return trust value

Figure 6: Fixing cold start using WikiTrust

The figure also shows an additional step (performed using Wikimedia API and described later in *Design*), which gets the page ID and the current revision id, needed for the WikiTrust API call.

The WikiTrust API that provides the needed data is called *Text Origin and Trust API*. It returns a special representation of a given revision of a page where each sequence (a single word or more consecutive words) has a trust value associated with it.

For example, in order to get the trust values of the revision 411787463 of the page 20742, we have to call:

```
http://en.collaborativetrust.com/WikiTrust/RemoteAPI?method=wik
imarkup&pageid=20742&revid=411787463
```

This call returns a representation like:

```
Since {{#t:10,84893431,User1}}its inception in
{{#t:8,86765634,User2}}1928 the movement
```

The tag `{{#t:10,84893431,User1}}` means that all the words following the closing bracket, in this case, `its inception in` has a trust value of 10 and it has been created by user `User1` in revision `84893431`.

In order to calculate the trust value for the whole document, we apply a weighted average, where the weight is given by the length of the sequence.

For getting the WikiTrust rating we therefore calculate:

$$\frac{\sum length(sequence) * trust(sequence)}{\sum length(sequence))}$$

Where:

`length(sequence)` = the length of the text sequence

`trust(sequence)`= the trust value of the text sequence

We however have to apply some changes in order to use the resulting WikiTrust rating.

One first change is adjusting the rating’s range to WRS, as WikiTrust ratings use a $[0,10]$ scale compared to the $[1,9]$ scale used by WRS.

We apply another change to the WikiTrust rating when we associate it with a category. Detecting the category of an article has been an interesting topic in WRS. Previous attempts of automatic classification for articles concluded with the use of a well-defined set of categories based on Open Directory Project classification scheme proposed by Pilkauskas (Pilkauskas, 2010). This classification scheme separates ratings in 15 categories and passes to the user the responsibility of assigning one of these categories to an article.

The ratings that WikiTrust provides for articles are content-based, therefore they are not assigned to any category. In order to overcome this issue, we perform the following steps:

1. When the WikiTrust rating is retrieved we present it to the user as it is, with no category assigned to it.
2. Whenever the user rates an article (rating involves selecting a category for it) the category of the WikiTrust rating will be updated to the user's rating category as described in *Figure 7: Updating the WikiTrust rating when the user rates the page.*

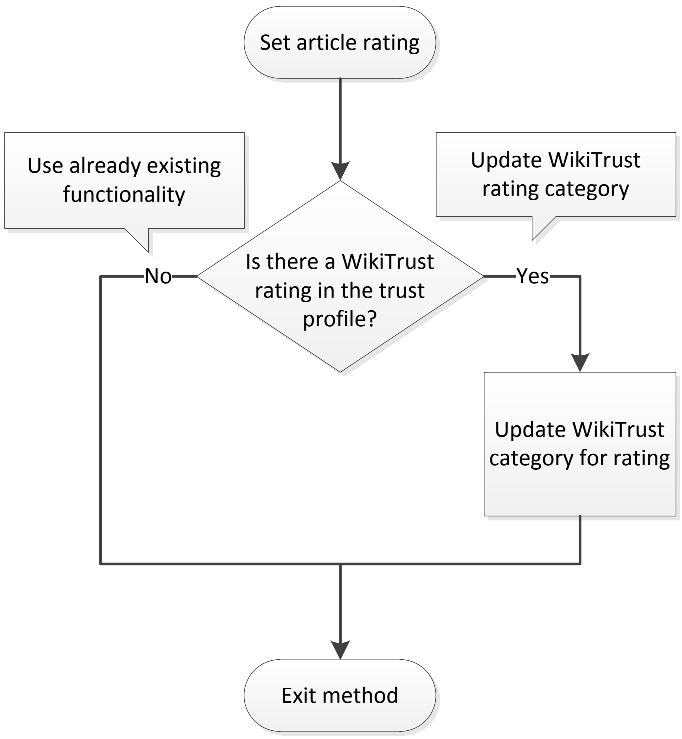

Set article rating
Use already existing functionality
Update WikiTrust rating category
No
Is there a WikiTrust rating in the trust profile?
Yes
Update WikiTrust category for rating
Exit method

Figure 7: Updating the WikiTrust rating when the user rates the page

Notice that assigning this category to a WikiTrust rating does not affect other users, as the WikiTrust rating being edited is personal and located in the user's trust profile.

3.1. Necessary System Architecture Changes

As illustrated in *State of the Art*, WRS faces several problems pointing to design changes.

The most important change is the location of the user ratings. After moving from the article page to user pages, the current approach of using Wikipedia as a central repository seems an impossible scenario. We will therefore look at other options for moving away from Wikipedia towards a central location for the ratings, even at the cost of having to maintain another online system. This change is mandatory as the current implementation does not work anymore.

Another change is the usage of Scone Proxy. The use of this framework slows down web browsing as all the content is bridged through Scone Proxy. Scone Proxy also has a series of restrictions which point to the fact that it might not be appropriate for a normal user. It conflicts with popular software as Skype and WAMP (Windows Apache MySQL PHP) Server. Another weakness of the Scone Proxy is that it doesn't follow the `no-cache` command in HTML and it caches pages. This means the user will have to pay attention to the output from the Scone Proxy Command Window and empty the cache manually, a task most users do not want to repeat whenever they return to a page they visited before. The last argument against using Scone Proxy is the installation of WRS. Even if the installation steps have been reduced over time, installing WRS is not an easy task as it requires some specific steps to be performed by the user in order to set up Scone Proxy. While this change is not mandatory, it will contribute to the development of WRS in the long run and will set the ground for wide distribution and usage of WRS.

The last motivation for change is the lack of advanced debugging provided by modern IDEs. The current implementation of WRS makes it hard to debug. As pointed out by (Pilkauskas, 2010) the limited way to debug the code is by dumping messages to either the console or log files or by a similar technique. The system does not benefit from the modern IDEs debugging tools which will otherwise enable a faster speed of development. Improving the debugging options while making the other changes will accelerate the current and future development. This change is optional, but highly necessary in the author's opinion.

After analysing the current state of the project which dictates several design changes (and keeping in mind the big goal of the thesis of solving the cold start problem) the following approach has been suggested:

Server Services

In order to get rid of Scone Proxy, the current functionality can be implemented as a service running on a server that exposes specific methods through web services. The goal is to keep the core of WRS unchanged and adapt it to fit the new architecture. Having these services will significantly reduce WRS installation complexity and it will change the distribution method, as such services can be used across platforms with virtually no limitations.

Independent Storage for Ratings

As the article ratings can no longer be kept on Wikipedia user pages, we have to move them to an independent location, accessible by the new WRS services. The server that holds these WRS services is a good candidate for this purpose as we can use it with no restrictions.

Currently, the ratings are being kept in plaintext in the user pages, and moving them to our own server would mean saving them in simple text files. Performing the supported operations (read, write) would translate to handling files on the server's file system. While these operations can be implemented, we have to be aware that their number, as well as their size, will be expanding consistently, along with the usage of WRS. In order to simplify things and focus on the main goal of this project, we turn our attention to already existing technologies for handling this scenario. We will therefore use a relational database, where all the needed operations are implemented and ready to be used with no or little extra work. Furthermore, the relational databases have already implemented mechanisms for fast lookup and transactional operations that we can take advantage of.

Browser Extension for Client Interaction

As all the complex operations will be executed by the server services, we can develop a lightweight client to access them. The client will have to be able to call web methods (exposed by the server services) and support a basic client interaction. These requirements, as well as the need of a simple and easy-to-distribute application, point towards implementing the WRS client as a browser extension. Ultimately, publishing this browser extension in an online marketplace as a free download will make it ideal for wide distribution and usage.

The current feedback could be improved as it is too intrusive, distracting the user from browsing. An interface where the article rating is presented without distracting the user from browsing, and displaying the feedback interface only on demand would be preferred to the existing one.

In the process of implementing this new browser extension, several improvements will have to be applied to the existing WRS Feedback interface in order to fix some of the problems it has in terms of user interaction. More specifically, the new interface should be less intrusive, more intuitive and optionally more esthetically appealing. This is an optional change, but it can contribute to a better user experience, which leads to attracting more users and keeping the existing ones.

An overview of the new proposed architecture can be observed in *Figure 8: New WRS Architecture*.

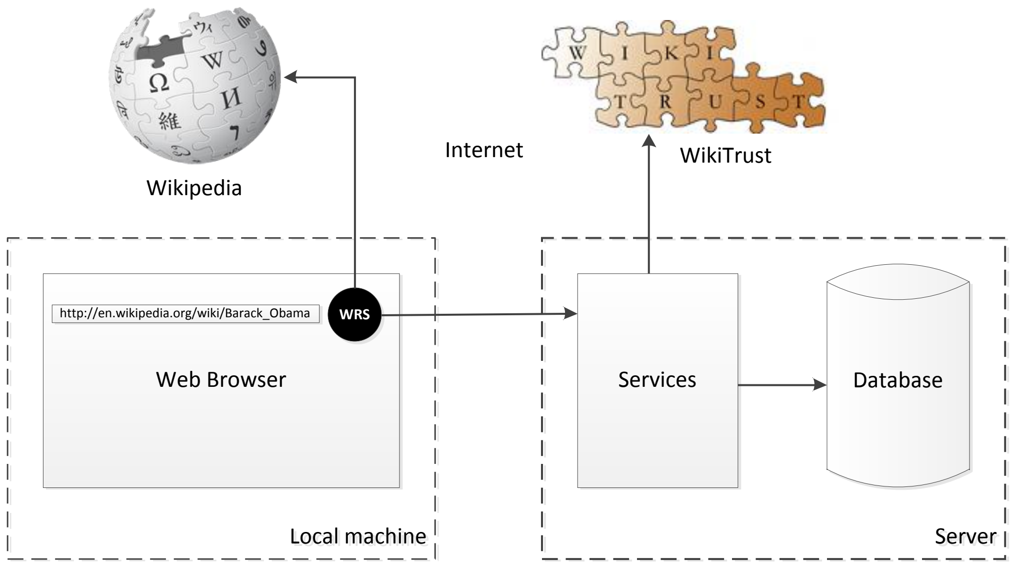


Figure 8: New WRS Architecture

Notice that we want to preserve the current core functions of the WRS (the ones that are dealing with trust management, trust dynamics etc.) and change only the adjacent components.

Implementing all these changes represent a major swift in the current WRS architecture. A much detailed plan for performing these changes is presented in *Design*, while in *Implementation* we take a closer look at the specifics of implementing the changes.

3.2. Summary

In this chapter we have discussed some of the mechanisms WikiTrust is using for building an article rating, which can be perceived as a reputation as it is the result of authors assigning trust in each other’s contributions through their own revisions. We have also looked at the differences between WikiTrust and WRS and how they can be changed in order for the two systems to work together.

In the second part of the chapter we have presented several architecture changes that will be implemented before the cold start problem can be addressed.

Chapter 4

4. Design

In this chapter we will discuss the new design of the system that is supposed to repair the issues we have found and accommodate the new changes needed in order to fix the cold start problem. Compared with the previous system, the new design that we are discussing represents a big change and a chance to address other problems that have not been stated in the goals of the project.

In this chapter we will have a detailed description of each core component of the new WRS.

4.1. Server Services

One of the most important changes in the current design is the location of the services performing the core functions of WRS. The previous design was running WRS on user's local machine in a decentralized manner. The new approach that we are taking is moving all the functions that were previously running on user's computer and move them to an independent server. There are several reasons for doing so, which have been stated in *Analysis*.

These services will perform all the heavy work of WRS, exposing as output only the results of the internal calculations. These services will be triggered by the users, through web services calls.

The web services will expose the following functionalities:

- Retrieve an article rating given the identity of the user.
- Allow a user to rate an article.
- Create a user account.

Internally, these services will inherit the functions from the old WRS system including handling the ratings and updating the trust profiles.

All the user data necessary for the system to function will be hosted on the same server, in the WRS database. The system will interact with both Wikipedia and WikiTrust in order to retrieve additional information needed for the cold start problem solution.

Scalability

The server services will be implemented in such a way as to easily enable future contributions. The idea is to create a scalable system, where developers can create modules for the operations currently supported. These modules will be detected and used without having to recompile or redeploy the services.

Therefore, the system will implement a plugin architecture, allowing additional modules to be developed, incorporated and executed at runtime. By using Java Reflection[23] these new modules will automatically be found by scanning the plugin directory and executed if needed.

The client using the server services will be responsible for specifying a plugin identifier on each method call, to help the server pick from the available modules able to execute a specific operation. If the client does not specify a plugin identifier, a default plugin (which we will implement) will be used. The default plugin will reuse the existing WRS functionality and will make the appropriate WikiTrust calls in order to handle the cold start problem.

[23] http://java.sun.com/developer/technicalArticles/ALT/Reflection/

4.2. Database

The key role of the database is to replace the Wikipedia user pages that have been used to store article ratings. The previous implementation kept these ratings in plaintext; therefore an additional parsing step is needed after reading them.

The easiest way to change the location of the ratings would have been to simply save the ratings to files on the server's file system in the same format. However, a better way to store the ratings would be in a relational database, which has obvious advantages over simple text files, the main important of which being strong typing (requiring no addition parsing after reading the data). Additionally, as a way of making the system more robust, the categories used for ratings will be moved into a database table, linked with the ratings table.

Our new approach relies on separate user accounts that are different from Wikipedia user accounts. The main reason for that is for ensuring privacy and data integrity. The user credentials will be kept in the database as well.

The database will also contain the (previously local) web of trust files. Each user has a web of trust file which is a basic serialization of the classes containing the data needed for calculating his trust values. Without changing much of the serialization, the location of that file will be moved in the database and linked with the user's table.

4.3. Browser Extension

Having a lightweight browser extension for WRS has been a goal starting since it early stages (Korsgaard, 2007). At that time, the development of such an extension was considered unfeasible mainly because it implied using specific programming languages, namely C++ or JavaScript which were inappropriate for accomplishing the tasks the old WRS system was performing.

In the context of the new architecture, the requirements for such an extension have changed, as all the heavy operations are done on a server.

Developing a browser has become possible and it has been chosen for reaching a larger audience and for providing an easy installation and little impact on the user's machine. Having a larger number of active users can also be the starting point of important collecting data for monitoring the way the system behaves and whether it lives up to the expectation.

The browser extension will take over the graphical user interface that was previously displayed to the user and will be the component responsible for invoking the server services through web services.

The browser extension will have the following features:

- Deliver minimum impact for the users while they are not browsing Wikipedia.
- Provide essential information when displaying a rating for a Wikipedia article in a less intrusive way than the previous WRS Feedback Interface.
- Allow the user to rate the currently viewed Wikipedia article.
- Allow the user to create accounts.

The browser extension will also have to take care of the user session management and perform all the communication with server services in a secure manner.

4.4. Security

The previous implementation, while having some security issues, relied on the security of the user's local machine as most of the sensitive data was handled only locally. However, with the new design we are suggesting, sensitive information will be handled both on the local machine as well as on the services server.

For the sake of simplicity and due to time constraints, the security model we will be using relies on the following assumptions:

- The client's browser is a secure environment and can handle sensitive data.
- The server which hosts the services is a secure environment.

Therefore, what we need to worry about is the communication channel. As a normal *http* connection when calling the web services could obviously be attacked, we have chosen to rely on the security of a secure *https* connection. As a consequence, our server will have to provide the means for establishing such a connection.

At the same time, as we mentioned before in *Database*, we choose to manage separate user accounts, different from the Wikipedia ones. The reason is that the user validation happens on the server, and by sending the user credentials to the server, if our server is corrupted it could reveal the user's Wikipedia credentials. By using the suggested separate user accounts, a corrupted server could at most modify or destroy the data used by WRS. In our opinion, this scenario causes less damage than potentially compromising one's Wikipedia account. We therefore have chosen to have a separate user management for your system.

When moving the WRS functionality onto the server we lose the privacy of user machines where the personal information has been kept. Even if we plan to use the same database for multiple purposes (public ratings, user management and personal trust profiles) access to these resources will be granted only to authenticated users and the data will only be accessible through the WRS web services. Therefore, no direct access will be provided to the WRS database.

4.5. Summary

In this chapter we continued the discussion we started in the *Analysis* with an in-depth look of the new WRS components. We set various requirements to be followed when implementing the system as a way to ensure the end result of these components working together meet the stated goals.

Chapter 5

5. Implementation

The existing WRS uses Java and other Java-related technologies to accomplish its goals. As we want to reuse as much code as possible, our new approach will have to be built in Java as well. The unneeded code will be eliminated while the code corresponding to the client side of the application will be rewritten in JavaScript.

5.1. Server Services

Several factors were considered when choosing the right approach to develop the server services. We had to accommodate the following requirements for the server services:

1. Interact with the database for data retrieval and manipulation.
2. Interact with external systems by using web clients.
3. Provide a convenient way for the browser extension to access the core WRS functionality.

By using a standard Java Web Application we could easily satisfy both *1* by using Java Persistence API[24] and *2* by using standard Java web clients. However, the Java Web Application only provides capabilities for creating standard web services. As we stated before, the browser extension will be interacting with these web services, therefore we want to avoid having to deal will the complex envelope design of the web service parameters and responses. We want to make the communication between any client and these services as easy as possible. Therefore, in order to satisfy *3*, we decided to use a special kind of web project, a Maven Web Application, which can use JSON [25] as a data format for web services communication.

JSON (JavaScript Object Notation) is a lightweight data format, similar to XML, which is the preferred way of data exchange on the web for scripting and lightweight programming languages. As we plan to use JavaScript for developing the Browser Extension, JSON data format is the perfect fit for client-server communication. An example of data representation using JSON can be observed in *Code snippet 1: JSON example*.

```
{"extensions": {
  "id": "wrs",
  "name": "WRS Chrome Extension",
  "UI": {
    "pages": [
      {"title": "feedback", "width": "200"},
      {"title": "options", "width": "220"}
    ]
  }
}}
```

Code snippet 1: JSON example

Our services will run on Glassfish Application Server[26], which is one of the obvious ways of hosting Java web services.

Entry Point

The place to start when analysing the server services is the `processRequest` method in `WRSResource` of the `wrs.web.resources` package. The signature of

[24] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html
[25] http://www.json.org/
[26] http://glassfish.java.net/

the method and the class can be seen in *Code snippet 2: processRequest method signature*. The method attributes are instructing the webserver to pass the *get* requests it receives when accessing the */wrs/* path to this method. You can also observe the attributes that instruct the server to produce JSON responses.

```
@Path("/wrs/")
public class WRSResource {
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public GenericResponse processRequest(…
```

Code snippet 2: processRequest method signature

After interpreting the input parameters, this method will call other functions depending on the `method` input variable. Several methods are currently supported:

- `createUser` creates an user account given an username and a password. e.g. /`method=createUser&username=mihai&password=somePassword`
- `login` checks if the provided username and password are valid. e.g. /`method=login&username=mihai&password=somePassword`
- `getCategories` returns the categories stored in the database. e.g. /`method=getCategories`
- `getRating` gets the rating of an article given the article URL and valid credentials. The returned response contains the rating value, the rating category and the percentage estimation of the rating category.
  e.g. a method call like:
  /`method=getRating&username=mihai&password=somePassword&pageUrl=http://en.wikipedia.org/wiki/Obama`
  can receive a response like
  `{"result":"success","rating":"7","categoryRatingPercentage":"90,00", "category":"2"}`
- `setRating` adds a new rating for the specified article given valid credentials, the article URL, rating value and rating category.
  e.g.
  `/wrs?method=setRating&username=mihai&password=somePassword&experience=true&rating=9&categoryRating=3&pageUrl=http://en.wikipedia.org/wiki`

Plugin Architecture

After deciding which method to call, the system looks at all the classes in a specific package that implement a specific interface (see *Code snippet 3: IResponseBuilder interface*). Based on an optional parameter `impl`, one of these classes will be selected for building the response message.

```
public interface IResponseBuilder {
    public int GetImplementationId();
    public RatingResponse GetRating(String pageUrl,String
username);
    public GenericResponse SetRating(String pageUrl,int
rating,int categoryId,boolean experience,String username);
}
```

Code snippet 3: IResponseBuilder interface

One important thing to notice is that the package is scanned at runtime by using Java Reflection (as illustrated in *Code snippet 4: loadClassesFromExternalPackage method*), therefore additional implementations can be added to the running system without recompiling or redeploying it.

```
public Class[] loadClassesFromExternalPackage() throws
IOException, ClassNotFoundException {
    ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
        assert classLoader != null;
        String packageName = "wrs.web.external";
        String path = packageName.replace('.', '/');
        Enumeration<URL> resources =
classLoader.getResources(path);
        List<File> dirs = new ArrayList<File>();
        while (resources.hasMoreElements()) {
            URL resource = resources.nextElement();
            dirs.add(new File(resource.getFile()));
        }
        ArrayList<Class> classes = new ArrayList<Class>();
        for (File directory : dirs) {
            classes.addAll(findClasses(directory,
packageName));
        }
        return classes.toArray(new Class[classes.size()]);
    }
    private static List<Class> findClasses(File directory,
String packageName) throws ClassNotFoundException {
        List<Class> classes = new ArrayList<Class>();
        if (!directory.exists()) {
            return classes;
        }
        File[] files = directory.listFiles();
        for (File file : files) {
            if (file.isDirectory()) {
                assert !file.getName().contains(".");
                classes.addAll(findClasses(file, packageName +
"." + file.getName()));
            } else if (file.getName().endsWith(".class")) {
classes.add(Class.forName(packageName + '.' +
file.getName().substring(0, file.getName().length() -
6)));//removes .class suffix
            }
        }
        return classes;
    }
```

Code snippet 4: loadClassesFromExternalPackage method

A Default Implementation

For the purpose of this thesis, a default implementation was developed for the plugin architecture, which provides the `getRating` and `setRating` functionalities using the proposed cold start problem solution.

These methods reuse some of the old code, adding on top the implementation for getting the WikiTrust rating when no rating is available in the system.

For improving the performance of the system, we update, in the `setRating` method, the WikiTrust rating category as illustrated in *Code snippet 5: ModifyWikitrustRating method* to the user rating category.

```
private void ModifyWikitrustRating(String pageUrl, int
categoryId) {
        Rating previousWikitrustRating =
webOfTrust.getRatingOfUserForPage(WikiTrustName, pageUrl);
        if (previousWikitrustRating != null) {
            previousWikitrustRating.setCategory(categoryId);
webOfTrust.insertNewRatingOfUserForPage(WikiTrustName, pageUrl,
previousWikitrustRating);
        }
    }
```

Code snippet 5: ModifyWikitrustRating method

Even if in the current implementation this step does not have any impact over the calculated value, (as WikiTrust rating is used only when no other ratings are available), keeping the WikiTrust rating in the user's trust profile and updating its category could be used in the future by other techniques that can be built on top of the current system to improve the cold start problem solution.

Data Access Layer

The classes in the package `wrs.web.dal` and `wrs.web.dal.tables` are used for interacting with database entities. For avoiding confusions with the previous terms used in the code the suffix *Table* has been added to the database entities and the class names have been capitalized. The mappings are presented in *Table 1: Entity mapping in wrs.web.dal.tables*.

SQL Table Name	**Web Application Entity Name**
rating	RatingTable
user	UserTable
category	CategoryTable
weboftrust	WeboftrustTable

Table 1: Entity mapping in `wrs.web.dal.tables`

`DatabaseInteraction.java` in `wrs.web.dal` provides methods for directly interacting with database entities to create, read, update and delete operations.

`DatabaseHelpers.java` in `wrs.web.dal` contains methods with auxiliary database functions. These methods typically use one or more methods from `DatabaseInteraction.java` for performing a more complex operation useful to WRS.

Trust Management

After being refactored for a better readability of the code, several classes from the old WRS implementation have been moved to `wrs.web.trust` and `wrs.web.rating` and distributed as follows

`wrs.web.trust`

- `Reviewer.java`
- `TrustUpdater.java`
- `WoT.java`

`wrs.web.rating`

- `InteractionData.java`
- `InteractionHistory.java`
- `Rating.java`
- `RatingCalculator.java`
- `RatingHistory.java`
- `SessionRatingDB.java`

These classes represent the main legacy from old implementation and their functionality has been maintained, with little changes.

One important change is in the `WoT.java` file. This file indirectly contains a reference to `InteractionData` class which has been modified from simply being an array wrapper to being a class containing appropriate fields and accessors. Because in the WRS implementation `WoT.java` is being serialized in order to store trust profiles, the changes that have been made are incompatible with older versions of class serializations. Typically, this situation should have been avoided, but given the fact that the code was hard to understand and the system was not used on a large scale, the changes have been adopted.

Helpers

Various static methods that are used in the project independently have been grouped in `wrs.web.helpers` package. This package contains methods for interacting with Wikipedia and WikiTrust, JSON and XML operations, as well as security and logging operations.

All needed Wikipedia operations have been moved into a single static class. These operations have been implemented using the latest MediaWiki[27] API. The following methods have been created:

- `ArrayList<Integer> GetLatestRevisions(String pageUrl, int revisionsCount)` returns the descending list of revisions given a pare URL.
- `int GetPageId(String pageUrl)` returns the ID of a page given its URL
- `String GetSanitizedArticleName(String articleName)` returns the name of the displayed article given an article name. This method is being used mainly for dealing with page redirects as it returns the actual article the user sees, which might be different than the one the address references.

WikiTrust offers a series of methods through its API; however we were only interested and have implemented a method for getting the trust value of the latest revision of the document:

- `getPageTrust (String articleUrl)` calls the *Text Origin and Trust WikiTrust API* in order to get a special representation where each word has a trust value associated. It uses the Wikipedia helper methods in order to provide specific parameters to the call and it then calculates the weighted document trust value which is used for fixing the cold start problem.

Dealing with Page Redirects

During development, certain scenarios have been found when WRS was not able to return a rating by using WikiTrust. The cause has been identified as being the way in which we used MediaWiki API and WikiTrust API for retrieving a rating.

Visiting the page on *Necessary and sufficient condition* at *http://en.wikipedia.org/wiki/Necessary_and_sufficient_conditions* (Notice the extra *s* at the end of the URL address) our extension uses MediaWiki API in order to retrieve the latest revisions. The article title used in the call is taken from the URL address, namely *Necessary_and_sufficient_conditions*. When doing so, only a three year old revision is received along with a comment saying the article was moved from *Necessary and sufficient conditions* to *Necessary and sufficient condition*. Attempting to use WikiTrust API for retrieving the page rating, using the provided revision, fails;

[27] http://www.mediawiki.org/wiki/MediaWiki MediaWiki is a PHP-based platform best known for being used by Wikipedia.

therefore, we end up in a scenario with no rating for the page, caused by the page redirect feature Wikipedia uses[28].

In order to fix this problem we have used another MediaWiki API call, that, when given an article name, it returns the redirected article, then one being displayed to the user.

Calling this method, it has therefore given us the correct article name and we have been able to retrieve recent revisions for which WikiTrust provided a rating.

Refactoring

The reused code from the previous system had to be refactored in order to make possible its understanding and its future development. A great deal of code was obscure and hard to understand, therefore hard to build upon. Therefore, as a first step, the existing code had to be refactored in order to ensure an implementation as close as possible to the initial one. The main focus was to build appropriate classes to replace the array wrappers classes used throughout the project and to make the code more understandable.

Unused Code

Multiple source code files containing unused functionality have been moved to `wrs.web.obsolete` package. This package also contains classes that are referenced in other classes, but not by any used ones. Their functionality has been therefore disconnected from the project, hence the decision of moving them to this package.

Other files that have been moved in this package are the ones mentioned by previous authors, but whose' functionality is no longer used in the project. We have not been able to track why or when these files stopped being used, and they have been simply ignored and moved to this package due to the complexity of analysing and integrating them into the new system.

[28] http://en.wikipedia.org/wiki/Wikipedia:Redirect

5.2. Database

A MySQL database containing four tables has been built in order to provide storage for all persistent data WRS uses:

- The *rating* table has replaced the old user pages where article ratings were stored. Notice the column *PageUrl* which represents the article URL. E.g. all ratings of article `http://en.wikipedia.org/wiki/Winter` which were before kept at `http://en.wikipedia.org/wiki/User:Recommendations/Winter` will now be placed in this table having `http://en.wikipedia.org/wiki/Winter` for *PageUrl* column.
- The *category* table is used to represent the WRS categories in the database. The *rating* table is linked to this table in order to associate a rating with a category.
- The *weboftrust* table stores the serialized user trust profile. The previous serialization functionality is kept, only the location of the data has been changed from the user's local machine file system, to the server's database.
- The *user* table contains the user credentials used for authentication. This table is linked with the *rating* table and *weboftrust* table.

The database schema can be observed in *Figure 9: Database schema*.

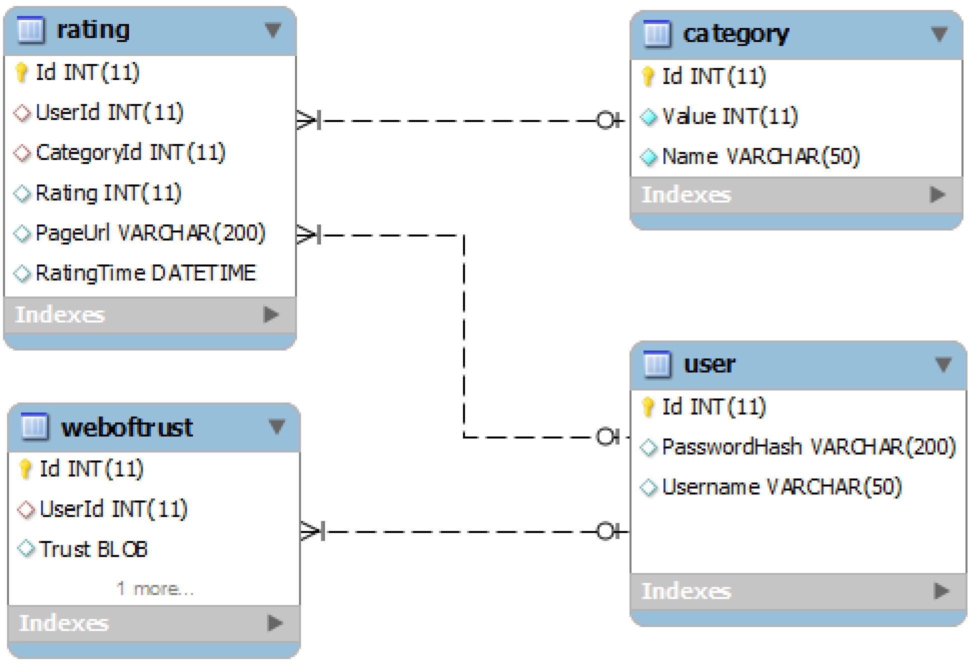


Figure 9: Database schema

5.3. Browser Extension

For implementing the browser extension, several popular browsers could have been chosen, as most of them provide an intuitive way of building extensions.

Because of the abundance of learning material and because it provides the needed functionality, the browser we have chosen is Google Chrome.

Google Chrome offers an easy framework for building extensions. Some of the features we have used while implementing the WRS Browser Extension are:

- Low impact on the navigation when the user is not viewing a Wikipedia article: we have chosen to implement the extension as a *page action* (the extension is accessible through a contextual icon placed on the navigation bar when visiting targeted websites) over a *browser action* (the extension is accessible through an icon placed on the browser toolbar that is always visible). We therefore have no impact when the user is visiting other pages different than Wikipedia as can be seen in *Figure 10: WRS Extension when visiting Wikipedia or a different page*.

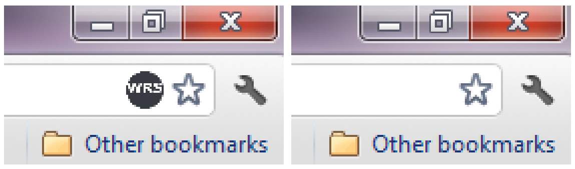
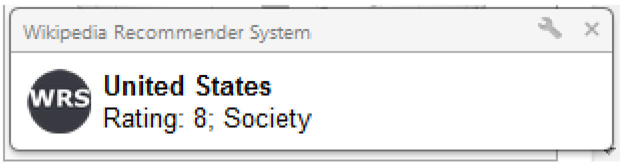


Figure 10: WRS Extension when visiting Wikipedia or a different page

- A notification system that is not intrusive and can display essential WRS information: we used the Chrome notification system that can display notification in the lower right corner of the screen. We show the notifications asynchronously, without any impact on Wikipedia page being loaded, and of course, only on relevant pages that belong to Wikipedia. An example of WRS Browser Extension notification can be seen in *Figure 11: WRS Browser Extension* notification. Any notification is closed automatically after 15 seconds and they can be turned off altogether from the *options* page.

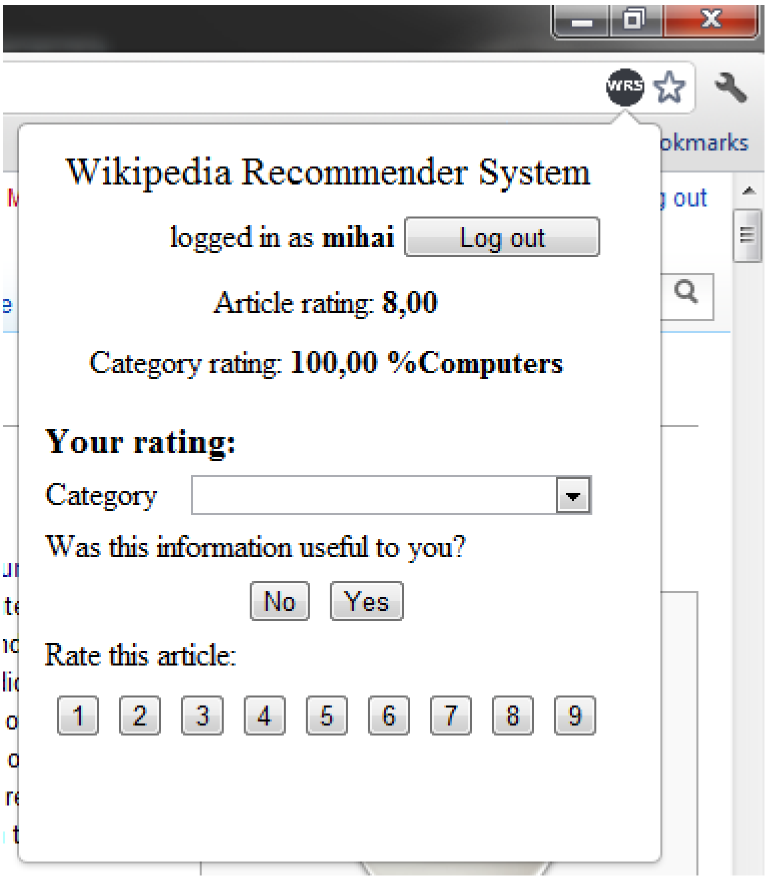


Wikipedia Recommender System

United States
Rating: 8; Society

Figure 11: WRS Browser Extension notification

- A popup window available on demand used for rating articles: when the user wants to rate the page he has to click on the navigation bar icon in order to open a popup window. Even if with this approach the user has to remember to rate the article in case he wants to, we think it is a better approach than displaying the window that allows rating at all times which blocks the navigation. The popup is only displayed when requested and disappears as soon as the user clicks anywhere else on the page. An example of popup window is shown in *Figure 12: WRS Extension popup window*.



Wikipedia Recommender System

logged in as **mihai** Log out

Article rating: **8,00**

Category rating: **100,00 %Computers**

Your rating:

Category

Was this information useful to you?

No Yes

Rate this article:

1 2 3 4 5 6 7 8 9

Figure 12: WRS Extension popup window

- A configuration page: Google Chrome allows an *options page* to be specified for extensions where the user can setup various parameters. Currently WRS Browser Extension is using the configuration page for multiple purposes among which:
    - Creating an user account
    - Authenticating an existing user
    - Turning on and off the article rating notifications

The WRS Extensions can make a successful rating retrieval for an article only after the user is logged in. The extension keeps the user credentials in the *localStorage* variables and transmits them over a secure *https* channel. We therefore rely on the browser's *localStorage* security which is acceptable as we have assumed from the start the user's local machine is a secure environment.

The extension has several components used to achieve the current behaviour:

- A *background page* is loaded when the browser starts. This page is used for deciding if a visited page is a valid Wikipedia article, therefore initializing an asynchronous call to the server services for retrieving the rating of the article (the *getRating* server method).
- A *popup page* displays the article rating and provides an interface for rating. This page therefore calls both the *getRating* and *setRating* server methods. The page can be opened by clicking on the extension icon in the navigation bar.
- An *options* page is used for logging in, creating a user account and changing the notification options. A screenshot of the notification page can be seen in *Figure 13: WRS Browser Extension options page*.

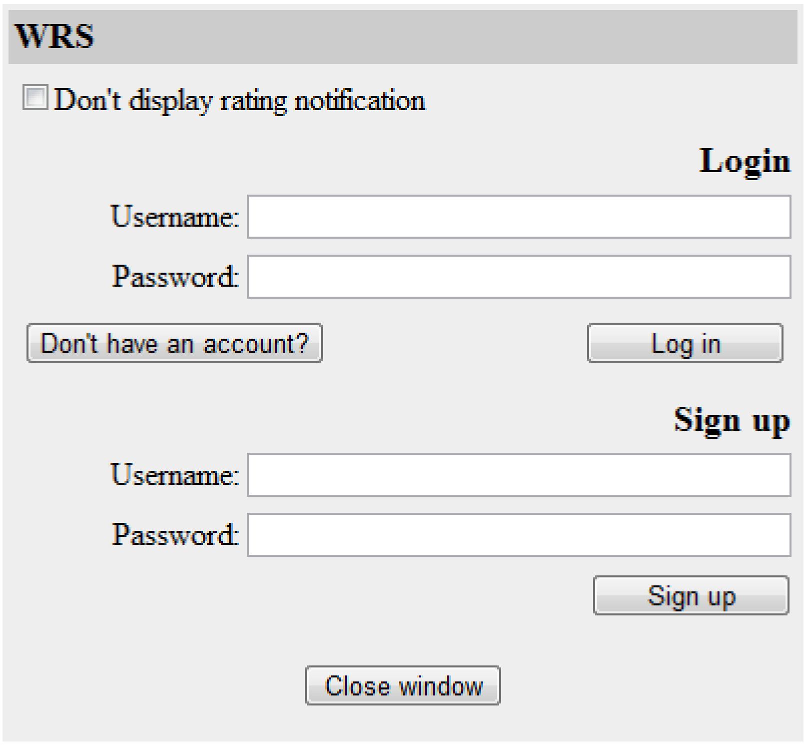


Figure 13: WRS Browser Extension options page

An interesting aspect of the built Browser Extension is the fact that all methods it invokes for interacting with the server services are cross domain calls. Typically, browsers do not allow this kind of client calls because of the potential security risks, but in the case of extension, Google Chrome provides a mechanism for bypassing this security measure. Extensions can specify a list of external domain list for which all the cross domain calls will be successful. These domains are presented to the user when he installs the extension. WRS Chrome extension asks for permission to communicate with *vmwrs.imm.dtu.dk*, the location of the web server hosting the WRS server services.

While the current WRS Extension relies on some specific Google Chrome features, it is our belief that it can easily be migrated to other browsers, especially to Mozilla Firefox because of the limited number of operation and the common programming language (JavaScript) used.

Publishing the Extension to Chrome Web Store

The next step of truly taking advantage of the browser extension development has been to publish it over the internet in a public location so that it can easily be installed and used.

Google Chrome has a large marketplace called Chrome Web Store that offers various browser extensions, themes, and applications that can be used inside the browser. The developed extension has been published in this online marketplace.

The current implementation of the WRS Chrome extension can be downloaded for free from the Chrome Web Store[29].

Current Limitations and Workarounds

In order the use *https* secure communication, the user of the extension has to accept the default Glassfish self-signed certificate of the server hosting the WRS services. However, because of the manual steps involved (described in detail in *Accepting the Self-signed Glassfish Certificate*), the published Chrome Extension uses the unencrypted *http* communication.

[29] Wikipedia Recommender System - Chrome Web Store
https://chrome.google.com/webstore/detail/dlbpjdiahnhhokdbanadnhgjfoiojdmb

This workaround can easily be changed by purchasing a paid SSL certificate for the server that requires no further manual steps in order to be accepted by the browser.

However, because of the academic nature of this project the default self-signed Glassfish certificate has been used. The decision of using an unsecure channel for communication was based on the need to simplify the usage of the WRS Chrome Extension and to spare potential users the trouble of manually installing the self-signed certificate.

5.4. Summary

In the current chapter we described the specifics of implementing the cold start problem solution while changing the system's architecture. We presented the details of multiple core components of the system as the Database, the Server Services and most importantly, the WRS Google Chrome Extension.

The Database has been introduced to replace the previous approach which kept the WRS data on the Wikipedia user pages. It consists of four tables created using MySQL database engine.

The Server Services have been built as a Maven Java Web Application and they represent the core of the WRS. The project uses Java Persistence API to interact with the Database. The Web Application also provides a way for clients to interact with WRS through web services.

The Google Chrome Extension was developed and published in the Chrome Web Store as a way to easily distribute WRS. This browser extension uses the publicly exposed Server Services methods to access the core functionality of WRS.

Chapter 6

6. Evaluation

The goal of this thesis is to fix the cold start problem in the WRS by using a content based filtering system. While making the changes for accomplishing this goal we have faced problems in storing the data used by the system on the Wikipedia user pages. Because of that, we had to redesign the system in order to address these challenges while still aiming to accomplish our goal of fixing the cold start problem. As a solution, we have moved both the core services and the rating storage on a separate server that can be used with no restrictions.

In this chapter we will look at some of the benefits that we have gained by changing the WRS architecture and implementing the cold start problem solution based on WikiTrust.

6.1. Contributions

In this section we will enumerate the achievements of the current project followed by a more in-depth discussion about each of them. The results of the current project are:

- A working WRS in the context of the changed Wikipedia policies.
- A working solution for the cold start problem in WRS.
- A working browser extension that enables easy installation and usage and large distribution of WRS.

6.2. Centralized versus Decentralized

The obvious change introduced by the new system design is the location of the core functionality of WRS. The previously version of WRS was a decentralized system meaning that each user was responsible for doing his own calculation and the data used by the project was stored locally. Only the common data that can be seen by everybody was kept in a central location i.e., the Wikipedia user pages. This approach had several advantages:

- *Scalability*: the individual calculations were performed on user’s machine therefore allowing a multitude of users to be able to run the system at the same time with virtually no performance penalty.
- *User data security*: the individual data was always kept on the user’s local machine, which is considered a safe environment. Keeping the WRS trust values private is obviously important.

On the other hand, the system encountered some problems like:

- *Application size and lack of an automatic installer*: the application was hard to install even for an advanced PC user and its core components conflicted with popular software mostly likely to exist on users machines. At the same time, for using the system, several changes had to be made to the browser configuration each time the user was switching the system on and off.
- *Distribution*: the application was hard to distribute and had a limited number of users because of that.

The current implementation has moved the core of WRS from the user’s machine to a separate server. Hence the system is a centralized one. The old system strengths are now the new system’s weaknesses:

- *Scalability*: as it gains more users, the system will be slowed down as it has to serve multiple requests. Nowadays, there are multiple ways to cope with this problem, but for the purpose of this thesis we have not looked at improving this aspect. We just acknowledge the fact that it will become a problem when the number of users increases.

- *User data security*: In the current implementation we keep the user data on the server, an environment that we consider secure, but which has not been completely analysed for security flaws. One measure that we have taken in this regard was to move away from using Wikipedia credentials to a separate user management system. This way, if the data is being breached, we are at least sure that data it is only relevant to WRS and not to any other system.

The new system's advantages are the following:

- *Application size*: The client side WRS application has a negligible size (under 50 KB) and does not conflict with any other software (as it is a browser extension). The steps involved in installing it are straightforward and are easy to perform even for a beginner user. In the current implementation, we do require one extra step for secure communication (accepting an SSL server certificate), that is a direct consequence of the academical nature of this project. However, the published extension uses unencrypted communication therefore no additional steps are required.
- *Distribution*: The current WRS Extension can be distributed using Chrome Web Store, potentially reaching more than 20% of the internet users[30]. The results of the current project have culminated with the developed Google Chrome Extension being published in the Chrome Web Store. The extension can be found in the *Productivity* category of the *Extension* section and can be downloaded[31] and used for free.

While both architectures have advantages and disadvantages, in the current implementation we had to choose the centralized approach in order to satisfy the new Wikipedia policies that have been blocking the functionality of the old WRS system.

6.3. Performance

Because of the design change, a comparison between the two systems (the old implementation and the current one) is not entirely appropriate. However we must mention the performance recorded by Korsgaard (Korsgaard, 2007), Lefevre (Lefevre, 2009) and Pilkauskas (Pilkauskas, 2010). The old implementation of WRS

[30] According to http://statcounter.com/ Google Chrome exceeded the 20% global market share of Internet browsers in June 2011
[31] Wikipedia Recommender System - Chrome Web Store https://chrome.google.com/webstore/detail/dlbpjdiahnhhokdbanadnhgjfoiojdmb

took anywhere from couple of seconds to more than a minute for displaying a rating during which time the navigation was blocked.

In the current implementation, the rating for a page is requested from the server asynchronously without stopping or slowing down the user navigation. When the rating is received it is displayed as a desktop notification. We can therefore say the WRS Extension has zero impact on the browser navigation. An important aspect to remember is that we are not using the Scone Proxy anymore. Therefore we eliminated any additional overhead that has been caused by its usage.

Retrieving the ratings from WikiTrust has been tracked during the experiments we present in *Cold Start Problem.* For 300 articles the average retrieval time was found to be 2.62 seconds while the median was 2.40 seconds. This duration refers only to the WikiTrust call and does not count the additional time spent on text manipulation, which is ignorable (bellow two milliseconds).

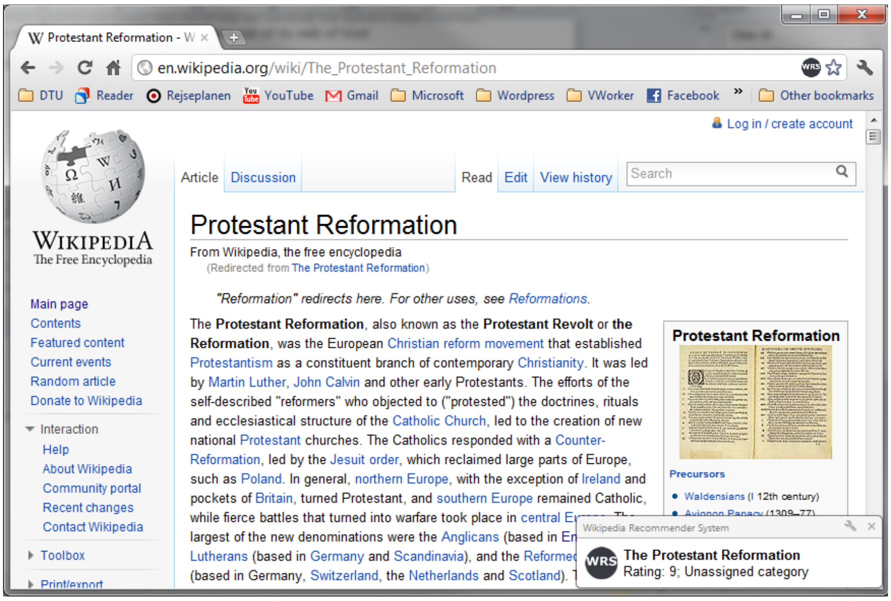
6.4. Cold Start Problem

The current goal of the thesis has been achieved, as currently the system uses WikiTrust ratings when it cannot calculate a trust value due to inexistent trust profiles.

This solution has been implemented in the following way:

- When the user visits a page he has never visited before, he receives a WikiTrust rating which is not assigned with any category, as seen in *Figure 14: WikiTrust rating used in WRS.*
- When the user rates a page, the WikiTrust rating is updated to the category the user selected for his rating. If the user revisits the page he will see the WikiTrust rating and the category he selected *Figure 15: WikiTrust rating associated with user's category*. Notice that the user rating does not influence the WikiTrust rating, and it is not considered when calculating the trust value for the article. This functionality has been inherited from the old WRS implementation.

An important thing to notice is that currently the WikiTrust ratings are kept in the user’s trust profile; therefore, other users cannot access it or use it directly.



Protestant Reformation - W ×

en.wikipedia.org/wiki/The_Protestant_Reformation

DTU Reader Rejseplanen YouTube Gmail Microsoft Wordpress VWorker Facebook » Other bookmarks

Log in / create account

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

Article Discussion Read Edit View history Search

Protestant Reformation

From Wikipedia, the free encyclopedia
(Redirected from The Protestant Reformation)

"Reformation" redirects here. For other uses, see Reformations.

The **Protestant Reformation**, also known as the **Protestant Revolt** or **the Reformation**, was the European Christian reform movement that established Protestantism as a constituent branch of contemporary Christianity. It was led by Martin Luther, John Calvin and other early Protestants. The efforts of the self-described "reformers" who objected to ("protested") the doctrines, rituals and ecclesiastical structure of the Catholic Church, led to the creation of new national Protestant churches. The Catholics responded with a Counter-Reformation, led by the Jesuit order, which reclaimed large parts of Europe, such as Poland. In general, northern Europe, with the exception of Ireland and pockets of Britain, turned Protestant, and southern Europe remained Catholic, while fierce battles that turned into warfare took place in central E
largest of the new denominations were the Anglicans (based in En
Lutherans (based in Germany and Scandinavia), and the Reforme
(based in Germany, Switzerland, the Netherlands and Scotland). T

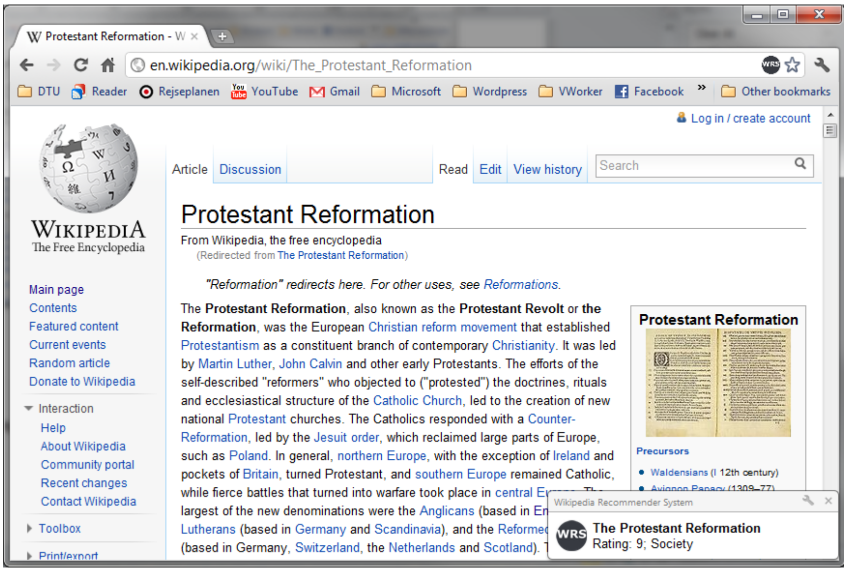
Protestant Reformation

Precursors
- Waldensians (l 12th century)
- Avignon Papacy (1309–77)

Wikipedia Recommender System

WRS The Protestant Reformation
Rating: 9; Unassigned category

Figure 14: WikiTrust rating used in WRS



Protestant Reformation - W ×

en.wikipedia.org/wiki/The_Protestant_Reformation

DTU Reader Rejseplanen YouTube Gmail Microsoft Wordpress VWorker Facebook » Other bookmarks

Log in / create account

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

Article Discussion Read Edit View history Search

Protestant Reformation

From Wikipedia, the free encyclopedia
(Redirected from The Protestant Reformation)

"Reformation" redirects here. For other uses, see Reformations.

The **Protestant Reformation**, also known as the **Protestant Revolt** or **the Reformation**, was the European Christian reform movement that established Protestantism as a constituent branch of contemporary Christianity. It was led by Martin Luther, John Calvin and other early Protestants. The efforts of the self-described "reformers" who objected to ("protested") the doctrines, rituals and ecclesiastical structure of the Catholic Church, led to the creation of new national Protestant churches. The Catholics responded with a Counter-Reformation, led by the Jesuit order, which reclaimed large parts of Europe, such as Poland. In general, northern Europe, with the exception of Ireland and pockets of Britain, turned Protestant, and southern Europe remained Catholic, while fierce battles that turned into warfare took place in central E
largest of the new denominations were the Anglicans (based in En
Lutherans (based in Germany and Scandinavia), and the Reforme
(based in Germany, Switzerland, the Netherlands and Scotland). T

Protestant Reformation

Precursors
- Waldensians (l 12th century)
- Avignon Papacy (1309–77)

Wikipedia Recommender System

WRS The Protestant Reformation
Rating: 9; Society

Figure 15: WikiTrust rating associated with user's category

For analysing the behaviour of the WRS module that handles the cold start problem we selected various types of articles. We expected these articles would show specific rating patterns.

The purpose of the experiments was to analyse the results of the WikiTrust ratings for known sets of articles. By observing those ratings and comparing them to the expectations we can draw conclusions about the solution we use for the cold start problem.

Featured Articles

The featured articles are considered by Wikipedia to have a comparable quality with the academic ones. On top of that, such articles accept only minor changes. As WikiTrust relies heavily on the age and stability of the changes, we were expecting high ratings on average for this experiment.

However, we have acknowledged that there are situations when WikiTrust is not able to retrieve an accurate rating for an article. Such a situation is encountered when expert authors get a low WikiTrust reputation, due to the lack of previous contributions. Articles written by such authors will most likely have an unfair low WikiTrust rating.

The articles used in this experiment have manually been selected from the list of featured articles on Wikipedia[32]. On this page, the articles are ordered alphabetically in categories, which in turn, are ordered alphabetically as well. The articles have been picked pseudo-randomly as we tried to have an equal distribution across the whole article set and we have visited articles only once.

Figure 16: WikiTrust rating distribution for featured articles and *Table 2: WikiTrust rating distribution for featured articles* show the results for visiting 100 featured articles.

98% of all articles have an above average rating (of 6 or more on our $[1,9]$ scale) and there are 0% articles with below average ratings. Furthermore, 76% of all articles have a rating of 8, which denotes a very high quality.

What this means is that the featured articles have indeed a high quality, and that the *featured* label, that Wikipedia gives to its best articles, indicates an above average quality.

[32] http://en.wikipedia.org/wiki/Wikipedia:Featured_articles

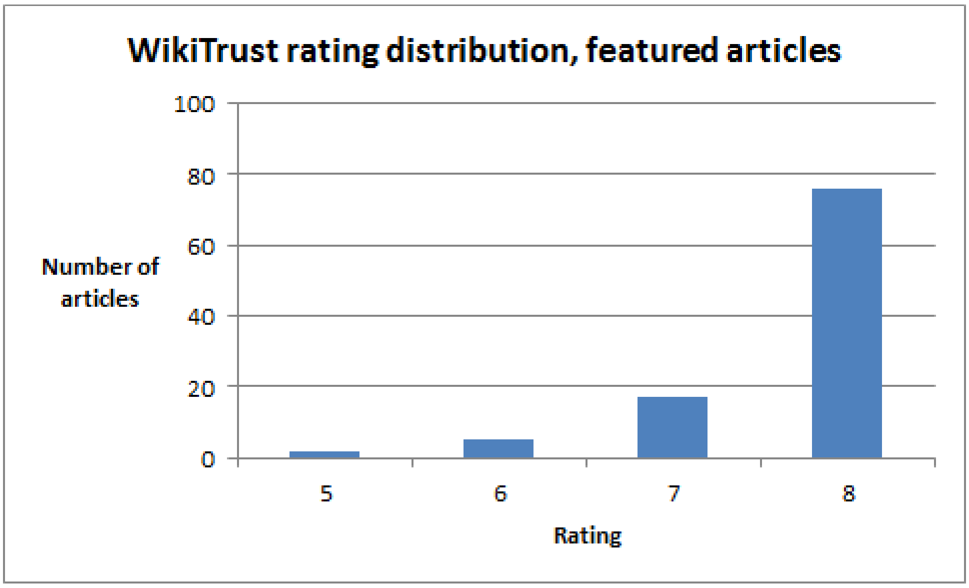


Figure 16: WikiTrust rating distribution for featured articles

Rating	Number of articles	Percentage
5	2	2.00%
6	5	5.00%
7	17	17.00%
8	76	76.00%
Total	**100**	**100.00%**

Table 2: WikiTrust rating distribution for featured articles

Poor Quality Articles

In this experiment we have handpicked various articles covering controversial, recent or local articles that are most likely to contain subjective or wrong information.

The articles considered controversial and recent are subject to numerous changes, which sometimes are made while the events take place. Such articles have been included in this experiment as the sources used for creating them are often not trustworthy (TV, radio) or biased (personal blogs).

The articles covering regional or local topics have few contributors, simply because few people know about them. These authors might contribute to Wikipedia only because they know something about the subject. Without support from the community or contributions to other articles, their reputation will most likely remain low. The articles they write will as well have a low quality.

Articles about new technologies and products might contain incomplete or wrong information resulting in poor quality.

We have acknowledged, however, that the articles analysed in this experiment can get wrong results (unfair low ratings or unmotivated high ratings) in the following situations:

- Articles might receive low ratings from WikiTrust when they contain recent changes and little information about the author. Over time, these articles might prove to have a high quality, but WikiTrust is not able to correctly estimate their quality shortly after they were written. This happens because WikiTrust relies heavily on the age of the text when it has little or no information about the author.
- Authors might build high reputation when their contributions are not changed, due to the lack of contributors. Such authors can build reputation over time and WikiTrust will incorrectly consider their contribution as having high quality.

The articles chosen for this experiment cover the following topics:

- On-going events (35 unique articles)
    - London riots (2011)
    - Norway attacks (2011)
    - Spanish protests (2011)
- Local/Regional topics from Denmark and Romania (47 unique articles)
- New technologies and products (18 unique articles)

The complete list of articles considered to have a poor quality can be found in *Appendix*.

Figure 17: WikiTrust rating distribution for poor quality articles and *Table 3: WikiTrust rating distribution for poor quality articles* summarize the results of visiting 100 supposedly low quality articles.

Notice that 48% of all articles have an above average rating (of 6 or more) while 26% of them have a below average rating (of 4 or less).

Notice that the rating distribution has changed: the featured articles experiment results are spread across the [5,8] range (with four steps) while the poor articles experiment results are spread across a wider, [2,7] range (with six steps).

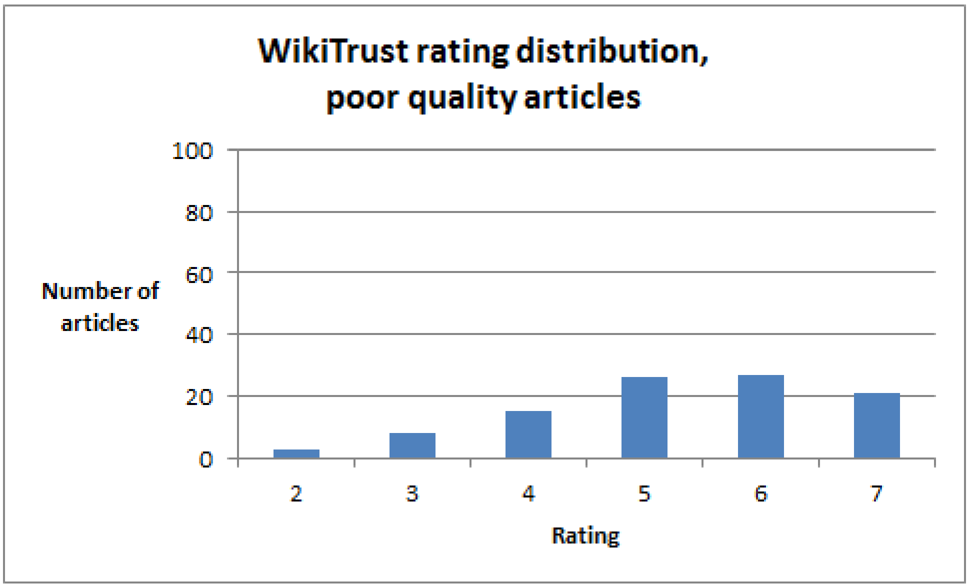


Figure 17: WikiTrust rating distribution for poor quality articles

Rating	**Number of articles**	**Percentage**
2	3	3.00%
3	8	8.00%
4	15	15.00%
5	26	26.00%
6	27	27.00%
7	21	21.00%
Total	**100**	**100.00%**

Table 3: WikiTrust rating distribution for poor quality articles

Random Articles

The articles visited for this experiment have been chosen pseudo-randomly, by a human operator. We have used several articles as starting points and then we have followed various links in the articles to navigate to other Wikipedia articles.

We notice that 89% of articles have an above average rating (of 6 or more) while 7% have a below average rating (of 4 or less).

The [2,8] rating distribution range contains seven steps and is wider than the ones presented in the previous experiments. This distribution resembles the featured article distribution and shows high results on average.

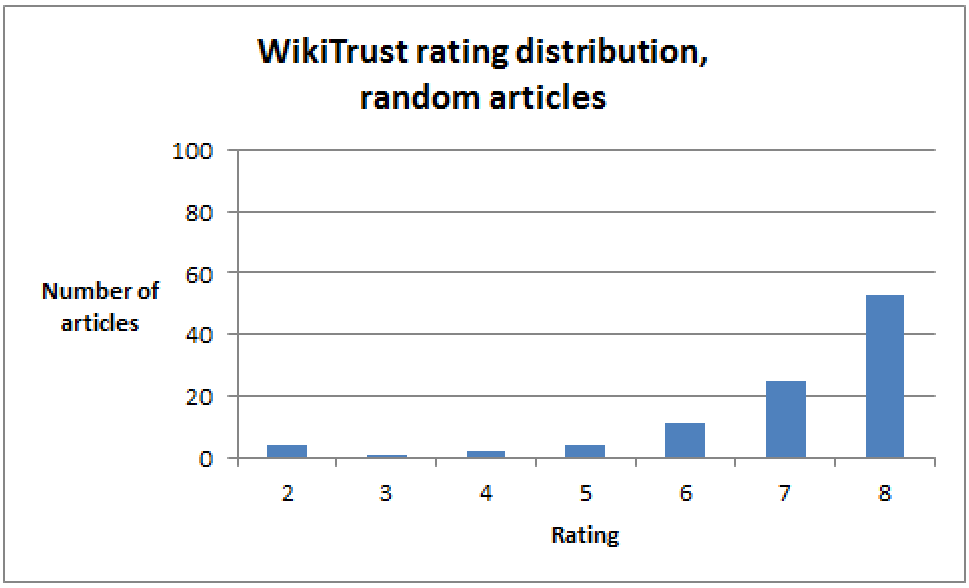


Figure 18: WikiTrust rating distribution for random articles

Rating	Number of articles	Percentage
2	4	4.00%
3	1	1.00%
4	2	2.00%
5	4	4.00%

6	11	11.00%
7	25	25.00%
8	53	53.00%
Total	**100**	**100.00%**

Table 4: WikiTrust rating distribution for random articles

Overall

The most important aspect of the experiments is the percentage of above average and below average ratings. There is a noticeable difference between featured articles and poor quality articles WikiTrust was able to detect and can be observed in (*Table 5: Experiments summary*). The random articles ratings are placed, as expected, in between the featured and poor quality articles when looking at both below average ratings and above average ratings.

Type of articles	**Below average rating (4 or less)**	**Average rating (5)**	**Above average rating (6 or more)**
Featured	0%	2%	98%
Poor quality	26%	26%	48%-
Random	7%	4%	89%

Table 5: Experiments summary

The small scale of the experiments we have performed and the various unknown (or hard to verify) factors involved makes it hard to draw precise conclusions. However, the fact that the results are the expected ones and that we have not got any conflicting or wrong results, leads us to believe WikiTrust is a useful and insightful tool for assessing the quality of Wikipedia articles based on their content.

Furthermore, based on the results, we consider the integration of WikiTrust into WRS is a good choice for our goal of fixing the cold start problem.

Figure 19: WikiTrust rating distribution contains all the results for the various types of articles visited.

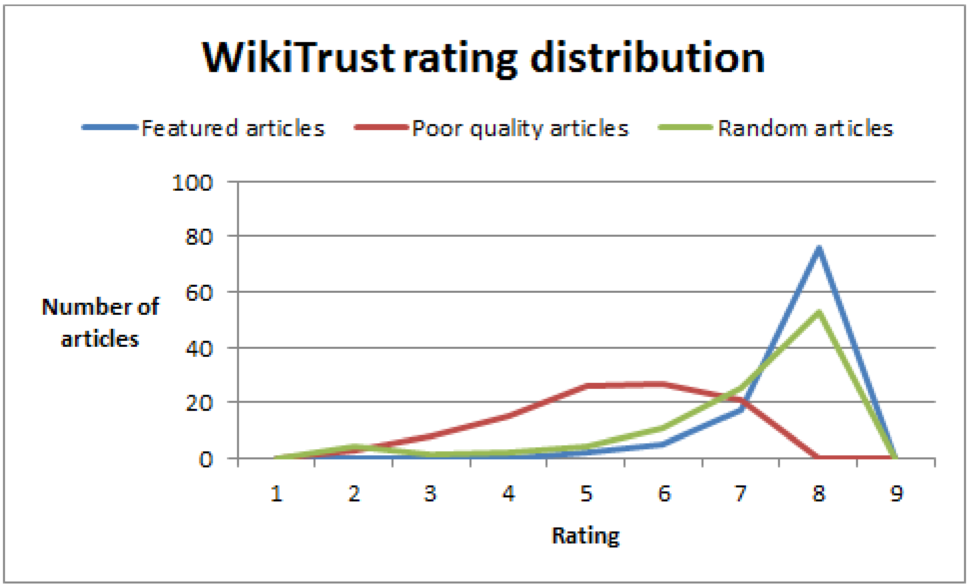


Figure 19: WikiTrust rating distribution

Notice that the figure does not show any ratings of 1 or 9, which are the extreme edges of the ratings range. Such examples were not found during the performed experiments. While individual sequences have extreme trust values, the overall rating for the article is averaged with all the other sequences, which most often contain various trust values. Articles with extreme ratings of 1 or 9 are likely to exist, but were simply not encountered during our experiments. As a solution, more experiments could be performed in order to find such articles or the WikiTrust ratings range could be adjusted to cover the WRS ratings range. However, in the current implementation, we are just acknowledging this situation, which can be the subject of further improvements.

6.5. UI & Other Improvements

One of the secondary goals of the project was to improve the WRS user experience. The new user interface was based on the previous one (illustrated in *Figure 20: Previous WRS Feedback Interface*), on top of which we added some modifications.

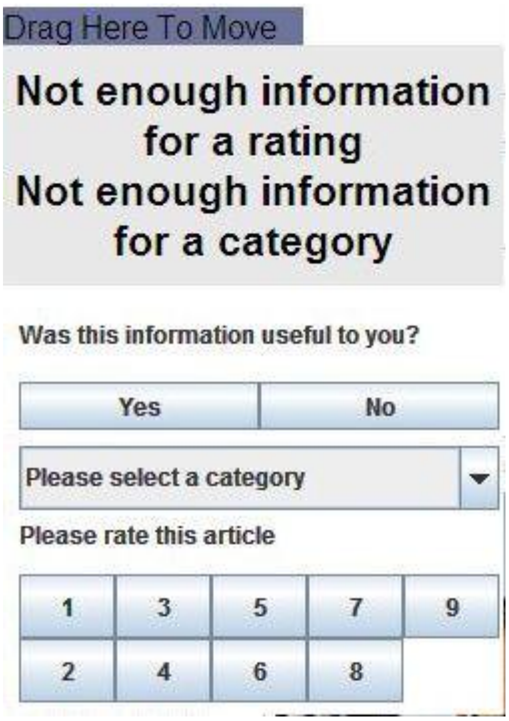

Drag Here To Move
Not enough information for a rating
Not enough information for a category
Was this information useful to you?
Yes
No
Please select a category
Please rate this article
1
3
5
7
9
2
4
6
8

Figure 20: Previous WRS Feedback Interface

The new system improved the client interaction in several ways:

- It displays a small notification window (300 pixels wide, 80+ pixels in height, depending on the length of the article title) containing essential information to the bottom right of the screen, which can be closes at any time and disappears after 15 seconds.
- It displays a popup window when clicking the WRS Extension icon on navigation bar. This popup has a similar size as the previous Feedback Interface, but it is only visible for the duration of the interaction, as it is being closed when the user clicks anywhere else on the page. As a browser extension, the new feedback mechanism integrates well with the pages and it does not present any rendering problems.
- The placement of the rating buttons has been changed to a clean left to right ascending order for a more intuitive way to rate an article.

The resulting feedback mechanism can be observed in *Figure 21: New feedback mechanism*.

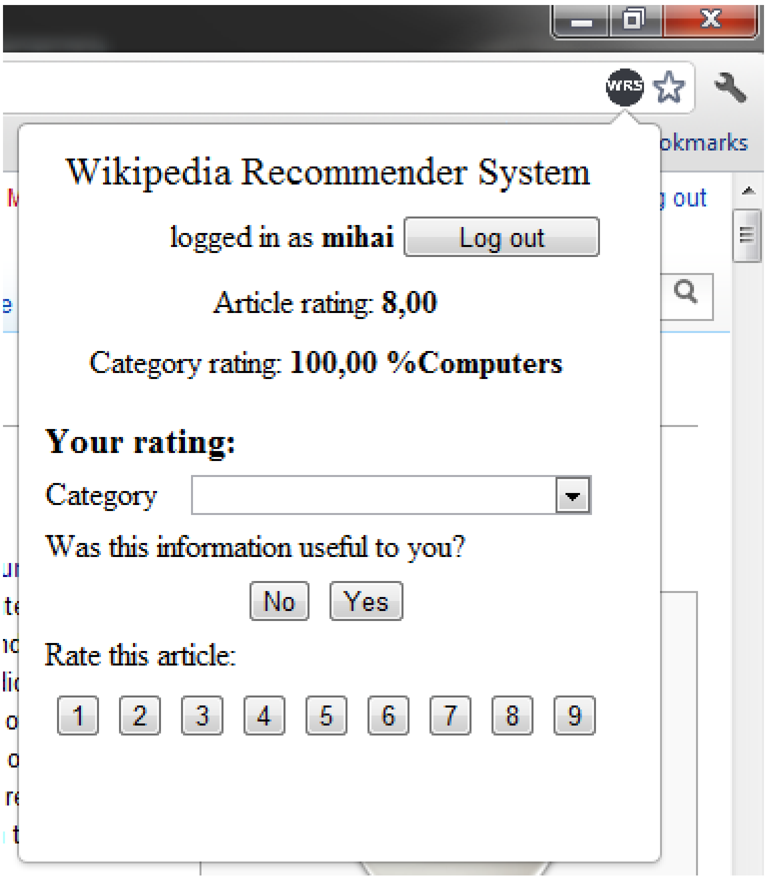


Figure 21: New feedback mechanism

From an installation and distribution perspective the system has improved radically. Any internet user can now download the browser extension that has been implemented and is available for free in the Chrome Web Store[33]. Finding the extension can be done either by:

- Direct link https://chrome.google.com/webstore/detail/dlbpjdiahnhhokdbanadnhgjfoiojdmb

[33] https://chrome.google.com/webstore

- Searching for a relevant term, part or the whole name of the extension by using the search functionality of the Chrome Store.

After finding the extension, the installation process only requires the user to confirm the permissions the extension needs. Currently, two permissions are requested as the extension needs to access browser tab information and the server hosting the WRS services. Once the permissions requests are accepted the extension is ready to be used, without even having to restart the browser.

Both of the presented methods for finding the extension and actually installing it should take under one minute and should be able to be performed even by a beginner PC user. From an installation and distribution perspective, we observe a tremendous improvement compared with the previous versions of WRS. We think that this aspect in particular will enable future work in analysing the benefits the WRS brings in order to further improve its functionality.

Several tweaks could be implemented in order to make the WRS browser extension a better productivity tool:

- Display the WikiTrust rating alongside the WRS rating, even after the cold start problem is not present. This modification could increase the perceived reliability by offering the two ratings, one content-based, the other one based on personal preferences.
- Present the number of ratings used in the trust calculation. This feature could increase user’s trust in the tool itself. Aside from the description about how the tool works, the current implementation does not allow the regular user to understand how the WRS rating was calculated. Displaying the users whose ratings were considered by WRS (or even more, displaying the reason why those users matter in the first place) could add to the credibility of the system.

These tweaks however have not been implemented due to time constraints.

6.6. Summary

In this chapter we have looked at various aspects of the WRS system after the changes, needed to accomplish the goals set in *Introduction,* have been implemented.

We have enumerated the changes between the old decentralized architecture and the current centralized one.

We have looked at the performance improvements and we have acknowledged the fact that they were mainly the desired consequence of the design change.

We have looked at the ways we addressed the Cold Start problem, the main goal of the project.

Finally, we have enumerated other ways in which the WRS was improved including numerous UI improvements and tweaks and the extension distribution using the browser’s online marketplace.

Chapter 7

7. Conclusion

7.1. Wikipedia Recommender System

In this project we have addressed the cold start problem present in WRS. The method chosen to achieve this goal was by using an external content-based filtering system, namely WikiTrust, in those situations where no user ratings are available.

We have presented the state of WRS and we have stated the need for design modifications due to the changes in Wikipedia policies, which were stopping us from using Wikipedia user pages as a repository.

We have analysed WikiTrust as an argument for being used in WRS due to its promising results.

We have enumerated the requirements for the new design of the system as a starting point for implementing the system.

A new WRS has been built around a new architecture that transforms the system into a centralized one. A browser extension has been implemented that accesses the centralized services. This browser extension has been published in the browser native marketplace, enabling easy installation and usage.

The evaluation of the new system has revealed several improvements in terms of performance, usability and distribution.

The implemented fix for the cold start problem has proved to be working, providing a rating for any article on Wikipedia. A set of tests have been performed in order to have an overview of the WikiTrust ratings. The results were the expected ones, therefore validating the idea of using WikiTrust in WRS in the first place.

7.2. Future Work

While accomplishing its goals, the current implementation has some areas that could be further improved.

First of all, if used on a large scale, a valid SSL server certificate should be purchased and the browser extension communication method changed to encrypted *https* mode.

Currently, the data in the WRS database is only accessible by authenticated users through our web services. Consuming these services, users can gain access to either public information like article ratings as well as to their personal trust profile. For ensuring privacy, a more advanced mechanism should be created for restricting access to private data, especially in the prospect of a larger and widely used system.

The communication security could be improved as well, as it currently relies on other systems' security, as the browser and server. Appropriate cryptographic means should be used in order to ensure the security of the system for the scenario where the above mentioned systems are compromised.

The previous authors of WRS stated the need for heavy code refactoring (Lefevre, 2009) in order to eliminate unneeded code and to improve the readability and scalability of the used one. In the current implementation this refactoring process has been started, but it has to be continued in future versions. More specifically, the new design implemented asks for data driven functionality opposite to various hard-coded values used in previous versions. Additionally, some work still needs to be done into transforming simple array wrappers classes into meaningful classes using fields and accessors for readability purposes.

7.3. Future Research

WikiTrust has a powerful API which can reveal interesting results about articles, authors and contributions. In the current project we have used *Text Origin and Trust API* in order to retrieve the current page rating. However, there are other APIs that can perform various operations, and an interesting one is providing a rating for an author. This function can be accessed by calling:

```
http://en.collaborativetrust.com/WikiTrust/RemoteAPI?method=raw
quality&revid=123
```

It would be interesting to observe what contribution (if any) the author rating has when using it to calculate article rating. Instead of using the trust value for the article (the current implementation), WRS could use this indicator as either a starting point for solving the cold start problem in a different way or for adding another dimension to the rating weighting.

Given the current architecture we think that building such tools and analysing their results is possible and can be used for bringing further improvements to WRS.

7.1. Summary of Conclusions

The project's goal of fixing the cold start problem in WRS has been achieved by using WikiTrust ratings.

In the process, WRS design had to be changed due to Wikipedia policies.

The outcome of this project is a working WRS system, with a working solution for the cold start problem.

Furthermore, the project can now be used by installing a free browser extension available in the Chrome Web Store.

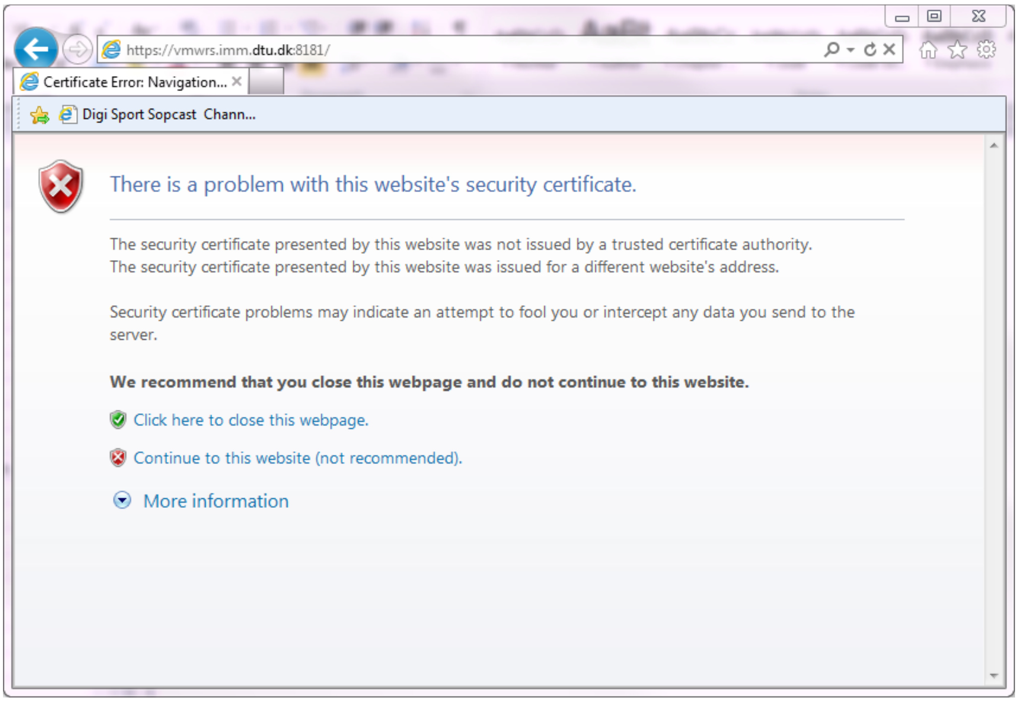
Chapter 8

8. Appendix

8.1. Accepting the Self-signed Glassfish Certificate

For using the secure *https* communication with the default self-signed Glassfish certificate, the following manual steps have to be performed:

1. Navigate to the *https* address of the website hosting the WRS services by using Internet Explorer. A Certificate Error message will be displayed, presenting two options as illustrated in *Figure 22: Certification error message for Glassfish's self-signed certificate*.


https://vmwrs.imm.dtu.dk:8181/
Certificate Error: Navigation...
Digi Sport Sopcast Chann...

There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

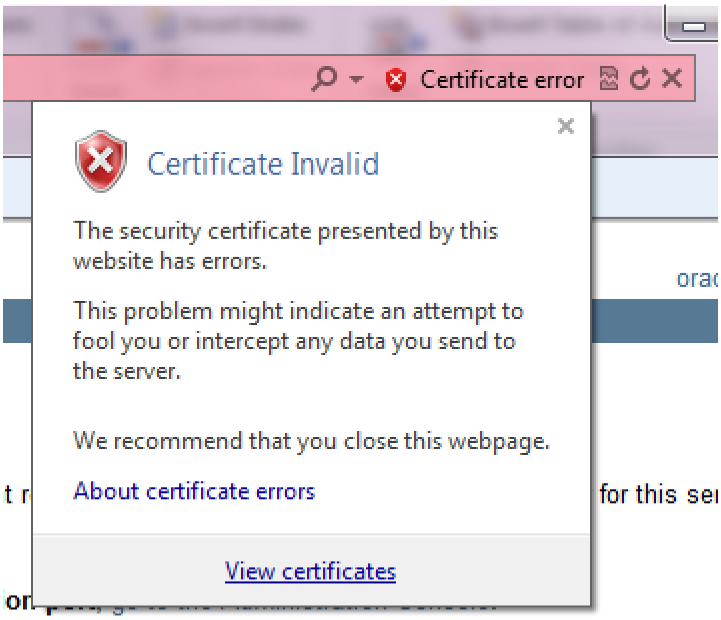
Click here to close this webpage.

Continue to this website (not recommended).

More information

Figure 22: Certification error message for Glassfish's self-signed certificate

2. Proceed to the website by selecting *Continue to this website (not recommended)* option.
3. Click on the *Certificate error* icon on the browser’s navigation bar and then click *View certificates* link at the bottom of the popup window (illustrated in *Figure 23: Certification error popup window*).


Certificate error

Certificate Invalid

The security certificate presented by this website has errors.

This problem might indicate an attempt to fool you or intercept any data you send to the server.

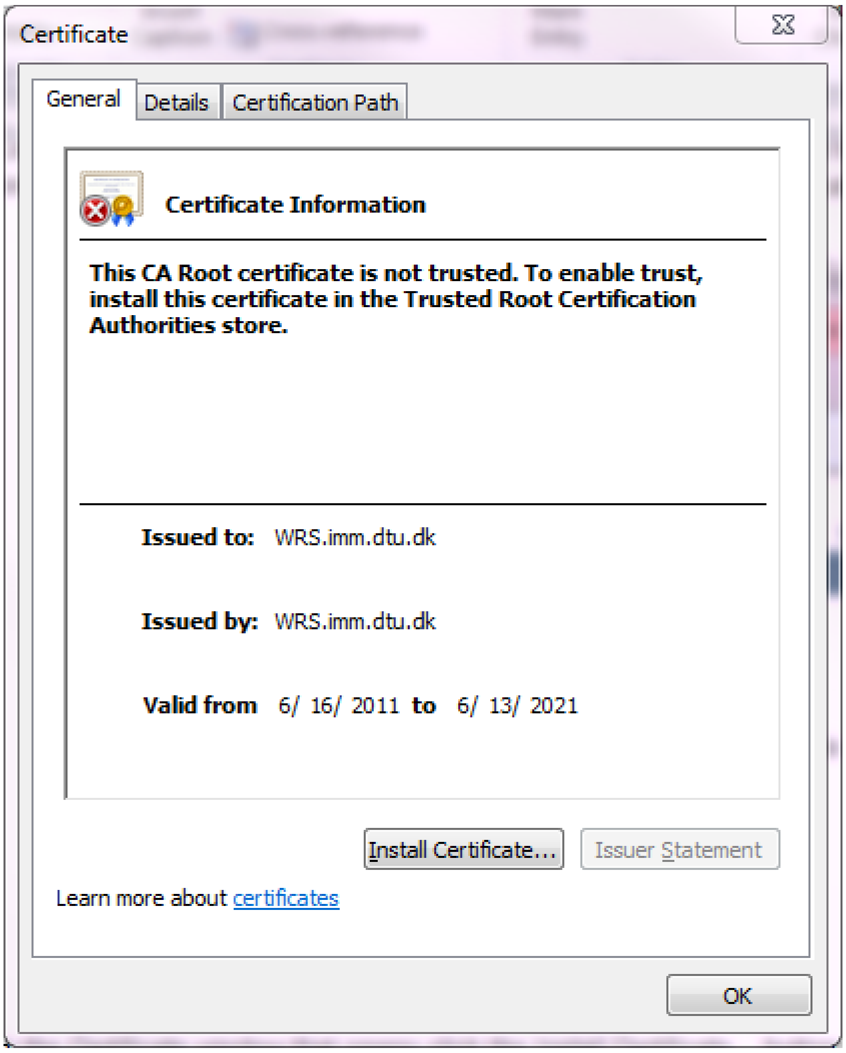
We recommend that you close this webpage.

About certificate errors

View certificates

Figure 23: Certification error popup window

4. In the *Certificate information* window that was opened click the *Install Certificate…* button (*Figure 24: Certificate information window*). This action will start the *Certificate Import Wizard*.



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: WRS.imm.dtu.dk

Issued by: WRS.imm.dtu.dk

Valid from 6/ 16/ 2011 **to** 6/ 13/ 2021

Learn more about certificates

Figure 24: Certificate information window

5. In the first step of the wizard select *Next*.
6. In the second step of the wizard select the *Place all certificates in the following store* option and click the enabled *Browse* button. This action will open the *Select Certification Store* window.
7. In the *Select Certification Store* window select *Trusted Root Certification Authorities* node and click the *OK* button.

8. Click the *Next* button to proceed to the third step of the wizard.
9. Finally click on the *Finish* button. This action will bring up a *Security Warning* asking you to confirm the requested operation. Click *Yes* to answer the dialog.
10. An information dialog will display the success of the operation which you can close by clicking the *OK* button. The *Certificate Information* window can now be closed as well.

8.2. Examples of WikiTrust Ratings

Featured Articles

Address	**Rating**	**Number of sections**	**Retrieval time (milliseconds)**
http://en.wikipedia.org/wiki/Daniel_Lambert	7	1640	2402
http://en.wikipedia.org/wiki/Green_children_of_Woolpit	7	850	2064
http://en.wikipedia.org/wiki/Ketuanan_Melayu	8	3399	2752
http://en.wikipedia.org/wiki/Hanged,_drawn_and_quartered	8	1605	2505
http://en.wikipedia.org/wiki/Postage_stamps_of_Ireland	8	1450	4028
http://en.wikipedia.org/wiki/The_Scout_Association_of_Hong_Kong	8	2206	2378
http://en.wikipedia.org/wiki/Taiwanese_aborigines	8	3284	3318
http://en.wikipedia.org/wiki/Encyclopædia_Britannica	8	3523	3407
http://en.wikipedia.org/wiki/Florida_Atlantic_University	8	2260	2942
http://en.wikipedia.org/wiki/Georgetown_University	7	3191	3856
http://en.wikipedia.org/wiki/The_Guardian_of_Education	8	696	2052
http://en.wikipedia.org/wiki/Ohio_Wesleyan_University	8	2730	3196
http://en.wikipedia.org/wiki/Plano_Senior_High_School	7	1681	2623
http://en.wikipedia.org/wiki/Royal_National_College_for_the_Blind	8	1415	2678
http://en.wikipedia.org/wiki/Some_Thoughts_Concerning_Education	8	1124	2230

http://en.wikipedia.org/wiki/Stuyvesant_High_School	8	2430	3547
http://en.wikipedia.org/wiki/Texas_Tech_University	8	2562	3529
http://en.wikipedia.org/wiki/Autostereogram	8	1188	2507
http://en.wikipedia.org/wiki/Matthew_Boulton	8	1765	2779
http://en.wikipedia.org/wiki/Construction_of_the_World_Trade_Center	8	1576	2483
http://en.wikipedia.org/wiki/Distributed_element_filter	7	1657	3203
http://en.wikipedia.org/wiki/Rolls-Royce_R	8	2351	2599
http://en.wikipedia.org/wiki/Scout_Moor_Wind_Farm	7	827	1938
http://en.wikipedia.org/wiki/Glynn_Lunney	8	784	2340
http://en.wikipedia.org/wiki/Colton_Point_State_Park	8	2035	2936
http://en.wikipedia.org/wiki/Covent_Garden	6	2079	3075
http://en.wikipedia.org/wiki/Craters_of_the_Moon_National_Monument_and_Preserve	8	1631	2716
http://en.wikipedia.org/wiki/Erie,_Pennsylvania	7	2207	2787
http://en.wikipedia.org/wiki/Germany	8	5076	3331
http://en.wikipedia.org/wiki/Hillsboro,_Oregon	8	2032	2928
http://en.wikipedia.org/wiki/Manchester	6	4612	3581
http://en.wikipedia.org/wiki/Paulins_Kill	7	1664	3020
http://en.wikipedia.org/wiki/Presque_Isle_State_Park	7	1475	2782
http://en.wikipedia.org/wiki/Forest_Park_(Portland,_Oregon)	7	1257	2278
http://en.wikipedia.org/wiki/Grand_Forks,_North_Dakota	8	2041	2891
http://en.wikipedia.org/wiki/Lethbridge	8	1582	3005
http://en.wikipedia.org/wiki/Pithole,_Pennsylvania	5	1233	2321
http://en.wikipedia.org/wiki/Turkey	7	4433	3269
http://en.wikipedia.org/wiki/Geology_of_the_Zion_and_Kolob_canyons_area	8	1494	2342
http://en.wikipedia.org/wiki/Chicxulub_crater	7	1202	2494
http://en.wikipedia.org/wiki/Geology_of_the_Grand_Canyon_area	8	1914	2690
http://en.wikipedia.org/wiki/Major_depressive_disorder	8	4094	2830
http://en.wikipedia.org/wiki/Multiple_sclerosis	8	2284	3451
http://en.wikipedia.org/wiki/Influenza	8	3410	3352
http://en.wikipedia.org/wiki/History_of_the_Grand_Canyon_area	8	1473	2113
http://en.wikipedia.org/wiki/History_of_Miami	8	1512	2696

http://en.wikipedia.org/wiki/Political_history_of_Mysore_and_Coorg_(1565–1760)	7	2601	2692
http://en.wikipedia.org/wiki/Political_integration_of_India	8	2276	2758
http://en.wikipedia.org/wiki/Scottish_National_Antarctic_Expedition	8	1100	2103
http://en.wikipedia.org/wiki/Manzanar	8	2587	2786
http://en.wikipedia.org/wiki/Samuel_Adams	8	2077	6443
http://en.wikipedia.org/wiki/James_Bowie	8	1683	2815
http://en.wikipedia.org/wiki/Tom_Crean_(explorer)	8	1383	3314
http://en.wikipedia.org/wiki/Thich_Quang_Duc	8	1535	2595
http://en.wikipedia.org/wiki/Epaminondas	8	1678	2537
http://en.wikipedia.org/wiki/Khalid_al-Mihdhar	8	1477	3228
http://en.wikipedia.org/wiki/Benjamin_Morrell	8	1095	3949
http://en.wikipedia.org/wiki/Fridtjof_Nansen	8	2133	3475
http://en.wikipedia.org/wiki/Emperor_Norton	8	1424	2836
http://en.wikipedia.org/wiki/Phan_Dinh_Phung	8	1288	3588
http://en.wikipedia.org/wiki/Ernest_Shackleton	8	2069	2886
http://en.wikipedia.org/wiki/Tarrare	7	515	1938
http://en.wikipedia.org/wiki/Nguyen_Chanh_Thi	7	2414	3214
http://en.wikipedia.org/wiki/Francis_Tresham	6	901	2148
http://en.wikipedia.org/wiki/Stephen_Trigg	8	645	1716
http://en.wikipedia.org/wiki/Hasekura_Tsunenaga	8	1873	2754
http://en.wikipedia.org/wiki/Harriet_Tubman	8	2134	2570
http://en.wikipedia.org/wiki/Roy_Welensky	8	967	2147
http://en.wikipedia.org/wiki/Chinese_classifier	8	2471	2537
http://en.wikipedia.org/wiki/Tamil_language	8	2443	2952
http://en.wikipedia.org/wiki/Swedish_language	8	3037	3120
http://en.wikipedia.org/wiki/Turkish_language	8	3812	3092
http://en.wikipedia.org/wiki/Edward_VIII_abdication_crisis	8	1448	2440
http://en.wikipedia.org/wiki/Fundamental_Rights,_Directive_Principles_and_Fundamental_Duties_of_India	7	1604	2446
http://en.wikipedia.org/wiki/Pendle_witches	8	1309	2175
http://en.wikipedia.org/wiki/Parliament_Acts_1911_and_1949	5	1449	2694
http://en.wikipedia.org/wiki/Toa_Payoh_ritual_murders	8	1562	2720

http://en.wikipedia.org/wiki/Australian_Competition_and_Consumer_Commission_v_Baxter_Healthcare	6	608	1941
http://en.wikipedia.org/wiki/Book_of_Kells	8	1906	2671
http://en.wikipedia.org/wiki/The_General_in_His_Labyrinth	8	2359	2813
http://en.wikipedia.org/wiki/La_Peau_de_chagrin	8	1089	2558
http://en.wikipedia.org/wiki/Harris's_List_of_Covent_Garden_Ladies	6	1030	2611
http://en.wikipedia.org/wiki/The_Story_of_Miss_Moppet	7	1228	2118
http://en.wikipedia.org/wiki/The_Time_Traveler's_Wife	8	1035	1940
http://en.wikipedia.org/wiki/Vijayanagara_literature_in_Kannada	8	2260	2623
http://en.wikipedia.org/wiki/Ode_on_a_Grecian_Urn	8	1836	2399
http://en.wikipedia.org/wiki/Chinua_Achebe	8	2419	3210
http://en.wikipedia.org/wiki/John_Day_(printer)	8	783	2332
http://en.wikipedia.org/wiki/Oliver_Wendell_Holmes,_Sr.	8	1817	2724
http://en.wikipedia.org/wiki/George_Moore_(novelist)	8	1015	2163
http://en.wikipedia.org/wiki/Elaine_Paige	8	2262	2345
http://en.wikipedia.org/wiki/Philitas_of_Cos	8	679	1886
http://en.wikipedia.org/wiki/Adelaide_Anne_Procter	8	705	1872
http://en.wikipedia.org/wiki/Ion_Heliade_R?dulescu	8	2215	2495
http://en.wikipedia.org/wiki/J._K._Rowling	8	2514	2761
http://en.wikipedia.org/wiki/Mary_Martha_Sherwood	8	1932	2282
http://en.wikipedia.org/wiki/John_Millington_Synge	8	921	1975
http://en.wikipedia.org/wiki/Mary_Wollstonecraft	8	2249	2841
http://en.wikipedia.org/wiki/0.999...	8	2429	2997
http://en.wikipedia.org/wiki/1_?_2_+_3_?_4_+_·_·_·	8	854	2217

Poor Quality Articles

Address	Rating	Number of sections	Retrieval time (milliseconds)

http://en.wikipedia.org/wiki/Death_of_Mark_Duggan	5	680	1794
http://en.wikipedia.org/wiki/2011_London_riots	5	2399	2652
http://en.wikipedia.org/wiki/Timeline_of_aftermath_of_2011_England_riots	2	443	1636
http://en.wikipedia.org/wiki/BlackBerry_Messenger	4	169	1289
http://en.wikipedia.org/wiki/Max_Hastings	7	364	5450
http://en.wikipedia.org/wiki/London	6	5684	4412
http://en.wikipedia.org/wiki/Norway_killings	6	2657	3112
http://en.wikipedia.org/wiki/International_reactions_to_the_2011_Norway_attacks	7	1529	2312
http://en.wikipedia.org/wiki/Regjeringskvartalet	5	137	1231
http://en.wikipedia.org/wiki/Anders_Behring_Breivik	7	2012	2768
http://en.wikipedia.org/wiki/Norwegian_Police_Service	5	738	1682
http://en.wikipedia.org/wiki/Utøya	6	277	1382
http://en.wikipedia.org/wiki/Labour_Party_(Norway)	7	715	1753
http://en.wikipedia.org/wiki/List_of_shooting_sprees	6	1049	1808
http://en.wikipedia.org/wiki/Tyrifjorden	5	201	1359
http://en.wikipedia.org/wiki/Beredskapstroppen	7	317	1294
http://en.wikipedia.org/wiki/Sigbjørn_Johnsen	3	111	1187
http://en.wikipedia.org/wiki/Øystein_Mæland	4	146	1278
http://en.wikipedia.org/wiki/Reactions_to_the_death_of_Osama_bin_Laden	6	4116	4112
http://en.wikipedia.org/wiki/Allegations_of_support_system_in_Pakistan_for_Osama_bin_Laden	6	392	1634
http://en.wikipedia.org/wiki/Asif_Ali_Zardari	5	4014	3747
http://en.wikipedia.org/wiki/Husain_Haqqani	6	343	1535
http://en.wikipedia.org/wiki/2011_Norway_attacks	6	2657	2590
http://en.wikipedia.org/wiki/Mayhem	5	119	1128
http://en.wikipedia.org/wiki/Al-Qaeda	7	5086	12037
http://en.wikipedia.org/wiki/Ayman_al-Zawahiri	7	1987	2406
http://en.wikipedia.org/wiki/Maadi	7	445	1477
http://en.wikipedia.org/wiki/2011_military_intervention_in_Libya	5	5222	4531
http://en.wikipedia.org/wiki/Al-Assad_Stadium	4	29	935
http://en.wikipedia.org/wiki/Hafez_al-Assad	7	934	1822

http://en.wikipedia.org/wiki/World_Youth_Day_2011	4	193	1411
http://en.wikipedia.org/wiki/Vanuatu	7	2182	3013
http://en.wikipedia.org/wiki/Bagdad	6	69	941
http://en.wikipedia.org/wiki/2011_Spanish_protests	7	1406	2319
http://en.wikipedia.org/wiki/Governance_of_the_Gaza_Strip	7	245	1313
http://en.wikipedia.org/wiki/Stadionul_Steaua	5	1030	2064
http://en.wikipedia.org/wiki/Romania's_Got_Talent	3	330	1621
http://en.wikipedia.org/wiki/Pro_TV	5	168	1315
http://en.wikipedia.org/wiki/Cronica_Cârcota?ilor	3	98	1158
http://en.wikipedia.org/wiki/Andreea_Marin_B?nic?	6	116	1119
http://en.wikipedia.org/wiki/Roman,_Romania	6	559	1507
http://en.wikipedia.org/wiki/D?muc	4	55	936
http://en.wikipedia.org/wiki/Bicazu_Ardelean	5	53	946
http://en.wikipedia.org/wiki/Bicaz_Canyon	6	76	911
http://en.wikipedia.org/wiki/National_Anticorruption_Directorate	5	37	912
http://en.wikipedia.org/wiki/High_Court_of_Cassation_and_Justice	5	81	916
http://en.wikipedia.org/wiki/DR1	4	229	1490
http://en.wikipedia.org/wiki/Got_Talent_series	7	2400	2598
http://en.wikipedia.org/wiki/Dansk_Melodi_Grand_Prix	5	415	1633
http://en.wikipedia.org/wiki/DR_HD	4	44	937
http://en.wikipedia.org/wiki/DR_Byen	3	88	970
http://en.wikipedia.org/wiki/Lucian_Bute	5	1050	2012
http://en.wikipedia.org/wiki/Leonard_Doroftei	4	917	1736
http://en.wikipedia.org/wiki/B.U.G._Mafia	7	1156	5077
http://en.wikipedia.org/wiki/Orice_E_Posibil	7	101	1898
http://en.wikipedia.org/wiki/JerryCo	6	175	1731
http://en.wikipedia.org/wiki/Via?a_noastr?_(Vol.1)	6	253	1451
http://en.wikipedia.org/wiki/Delia_Matache	5	276	1354
http://en.wikipedia.org/wiki/Connect-R	4	68	979
http://en.wikipedia.org/wiki/Kato_(Producer/Artist)	2	190	1313
http://en.wikipedia.org/wiki/Gazeta_Sporturilor	3	112	1116
http://en.wikipedia.org/wiki/Jurnalul_Na?ional	6	57	912
http://en.wikipedia.org/wiki/Antena_1_(Romania)	5	120	1137

http://en.wikipedia.org/wiki/Intact_Group	3	91	1124
http://en.wikipedia.org/wiki/Romanian_Footballer_of_the_Year	4	790	1660
http://en.wikipedia.org/wiki/Antena_4_-_Euforia_lifestyle_TV	2	37	897
http://en.wikipedia.org/wiki/Dan_Voiculescu	5	622	1864
http://en.wikipedia.org/wiki/Acas?	4	53	983
http://en.wikipedia.org/wiki/DDTV	5	29	936
http://en.wikipedia.org/wiki/Oglinda_TV	5	70	1040
http://en.wikipedia.org/wiki/Antena_Interna?ional	4	74	925
http://en.wikipedia.org/wiki/Antena_2_(Romania)	3	42	954
http://en.wikipedia.org/wiki/Antena_3_(Romania)	5	101	1109
http://en.wikipedia.org/wiki/Antena_2_(Romania)	3	42	925
http://en.wikipedia.org/wiki/Antena_Interna?ional	4	74	869
http://en.wikipedia.org/wiki/Cristian_Chivu	6	2040	2162
http://en.wikipedia.org/wiki/GSP_TV	5	125	1213
http://en.wikipedia.org/wiki/Liga_I	6	2920	2733
http://en.wikipedia.org/wiki/Realitatea_TV	6	203	1359
http://en.wikipedia.org/wiki/Romanian_Professional_Football_League	6	105	1250
http://en.wikipedia.org/wiki/FC_Universitatea_Craiova	7	484	1627
http://en.wikipedia.org/wiki/FC_Dinamo_Bucure?ti	7	1792	2351
http://en.wikipedia.org/wiki/Motorola	6	1366	2190
http://en.wikipedia.org/wiki/WebOS	7	1192	2091
http://en.wikipedia.org/wiki/Windows_Phone_7.1	6	268	1424
http://en.wikipedia.org/wiki/Steve_Ballmer	7	553	1829
http://en.wikipedia.org/wiki/Windows_Phone_7.1	6	268	1377
http://en.wikipedia.org/wiki/HP_TouchPad	4	569	1743
http://en.wikipedia.org/wiki/Google_Plus	6	593	2753
http://en.wikipedia.org/wiki/Firefox_6	6	4660	9088
http://en.wikipedia.org/wiki/Anonymous	6	134	1336
http://en.wikipedia.org/wiki/WebOS	7	1192	2562
http://en.wikipedia.org/wiki/Samsung_Galaxy_S_II	5	1350	2218
http://en.wikipedia.org/wiki/W00t	4	217	1669
http://en.wikipedia.org/wiki/Windows_8	6	480	2125
http://en.wikipedia.org/wiki/Ps_vita	5	992	3517

http://en.wikipedia.org/wiki/BlackBerry_Bold	6	492	3064
http://en.wikipedia.org/wiki/Samsung_Galaxy_Tab_10.1	5	740	1697
http://en.wikipedia.org/wiki/Android_(operating_system)	5	2496	2383
http://en.wikipedia.org/wiki/Ie10	7	323	1572

Random Articles

Address	Rating	Number of sections	Retrieval time (milliseconds)
http://en.wikipedia.org/wiki/Arizona	8	3830	7585
http://en.wikipedia.org/wiki/Scottsdale,_Arizona	7	2383	5867
http://en.wikipedia.org/wiki/Paradise_Valley,_Arizona	3	434	2077
http://en.wikipedia.org/wiki/Carefour	7	1333	3311
http://en.wikipedia.org/wiki/Wal-Mart	8	3028	4837
http://en.wikipedia.org/wiki/West_Plains,_Missouri	6	383	4586
http://en.wikipedia.org/wiki/The_Home_Depot	8	1111	2293
http://en.wikipedia.org/wiki/Cobb_County,_Georgia	8	1353	2087
http://en.wikipedia.org/wiki/Metro_Atlanta	8	3844	6202
http://en.wikipedia.org/wiki/Columbus,_Georgia_metropolitan_area	7	158	1266
http://en.wikipedia.org/wiki/Thanksgiving	8	730	2884
http://en.wikipedia.org/wiki/New_France	8	1360	5022
http://en.wikipedia.org/wiki/American_Revolution	8	3915	5248
http://en.wikipedia.org/wiki/Pilgrim	6	386	1606
http://en.wikipedia.org/wiki/World_War_I	8	7773	8296
http://en.wikipedia.org/wiki/Franklin_D._Roosevelt	8	4935	4891
http://en.wikipedia.org/wiki/Liberia	6	2577	3056
http://en.wikipedia.org/wiki/Magnets	8	2094	3104
http://en.wikipedia.org/wiki/Redox	7	1375	3192
http://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)	7	575	2183
http://en.wikipedia.org/wiki/Facade_pattern	6	171	1352

http://en.wikipedia.org/wiki/Object-oriented_programming	8	1589	4890
http://en.wikipedia.org/wiki/C++	8	1815	6817
http://en.wikipedia.org/wiki/Java_(programming_language)	7	2016	3626
http://en.wikipedia.org/wiki/Java_Servlet	7	551	1711
http://en.wikipedia.org/wiki/JSON	8	851	2008
http://en.wikipedia.org/wiki/JavaScript_syntax	7	1441	3476
http://en.wikipedia.org/wiki/First_Amendment_to_the_United_States_Constitution	8	1963	2421
http://en.wikipedia.org/wiki/FC_Barcelona	8	3362	2817
http://en.wikipedia.org/wiki/Athletic_Bilbao	8	2033	5423
http://en.wikipedia.org/wiki/Real_Madrid	6	2907	2505
http://en.wikipedia.org/wiki/Camp_Nou	8	812	1895
http://en.wikipedia.org/wiki/La_Masia	8	1416	2034
http://en.wikipedia.org/wiki/Ballon_d'Or	7	3297	2893
http://en.wikipedia.org/wiki/Andrés_Iniesta	8	1629	2273
http://en.wikipedia.org/wiki/Lionel_Messi	8	3662	3133
http://en.wikipedia.org/wiki/Xavi	7	1339	3644
http://en.wikipedia.org/wiki/El_Clásico	8	4681	4726
http://en.wikipedia.org/wiki/Catalan_nationalism	8	1257	2283
http://en.wikipedia.org/wiki/Autostrada_Soarelui	5	381	1364
http://en.wikipedia.org/wiki/?ód?	8	1798	5804
http://en.wikipedia.org/wiki/Warsaw–Vienna_Railway	7	237	1345
http://en.wikipedia.org/wiki/Congress_Poland	8	1047	2067
http://en.wikipedia.org/wiki/Austro-Hungarian_Empire	8	3638	3906
http://en.wikipedia.org/wiki/Polystyrene	7	1418	2270
http://en.wikipedia.org/wiki/Polyurethane	8	1643	3215
http://en.wikipedia.org/wiki/Carbamate	7	385	1461
http://en.wikipedia.org/wiki/Synthetic_fibers	7	368	1274
http://en.wikipedia.org/wiki/Hydroxyl	6	270	1437
http://en.wikipedia.org/wiki/Blowing_agent	2	104	4291
http://en.wikipedia.org/wiki/Ethylene_glycol	8	841	2050
http://en.wikipedia.org/wiki/Fenfluramine	7	179	1424
http://en.wikipedia.org/wiki/Dodge	8	1557	2537
http://en.wikipedia.org/wiki/Copenhagen	7	4214	4896

http://en.wikipedia.org/wiki/Christian_IV_of_Denmark	8	901	2065
http://en.wikipedia.org/wiki/List_of_most_expensive_cities_for_expatriate_employees	6	3872	4991
http://en.wikipedia.org/wiki/Forbes	8	868	2516
http://en.wikipedia.org/wiki/Metropolitan_area	8	877	5356
http://en.wikipedia.org/wiki/Water	8	3315	4658
http://en.wikipedia.org/wiki/H20	5	39	912
http://en.wikipedia.org/wiki/Iridocyclitis	6	164	1185
http://en.wikipedia.org/wiki/Over_the_counter	8	655	1661
http://en.wikipedia.org/wiki/Active_pharmaceutical_ingredients	5	180	1278
http://en.wikipedia.org/wiki/Chris_Spheeris	4	99	1128
http://en.wikipedia.org/wiki/Greek-American	8	1317	6794
http://en.wikipedia.org/wiki/New_York	8	2699	4697
http://en.wikipedia.org/wiki/Kingdom_of_Great_Britain	7	1087	3907
http://en.wikipedia.org/wiki/Kingston,_New_York	7	1058	2773
http://en.wikipedia.org/wiki/Ebola	8	1692	3463
http://en.wikipedia.org/wiki/Yellow_fever	8	1325	3029
http://en.wikipedia.org/wiki/Bleeding_diathesis	4	301	1563
http://en.wikipedia.org/wiki/Coagulopathy	6	82	1153
http://en.wikipedia.org/wiki/Hypercoagulability	7	1078	3047
http://en.wikipedia.org/wiki/Antithrombin_deficiency	6	124	1444
http://en.wikipedia.org/wiki/Recessive	2	228	2165
http://en.wikipedia.org/wiki/Gregor_Mendel	7	822	2462
http://en.wikipedia.org/wiki/Austrian_Empire	8	1013	2360
http://en.wikipedia.org/wiki/Silicon_Valley	8	1448	2871
http://en.wikipedia.org/wiki/IPad	8	2767	3890
http://en.wikipedia.org/wiki/Italy	8	5801	5824
http://en.wikipedia.org/wiki/IPhone	8	3973	3753
http://en.wikipedia.org/wiki/Cellulosic_ethanol	8	1568	2513
http://en.wikipedia.org/wiki/Wood_pulp	7	821	1842
http://en.wikipedia.org/wiki/Logging	7	445	1500
http://en.wikipedia.org/wiki/Cut-to-length_logging	7	138	1222
http://en.wikipedia.org/wiki/Skidder	8	256	1263
http://en.wikipedia.org/wiki/Four_wheel_drive	8	2863	2929

http://en.wikipedia.org/wiki/Off-road_vehicle	8	752	1845
http://en.wikipedia.org/wiki/Desert_Racing	5	91	1063
http://en.wikipedia.org/wiki/Baja_1000	8	1267	2075
http://en.wikipedia.org/wiki/La_Paz,_Baja_California_Sur	6	528	2839
http://en.wikipedia.org/wiki/La_Paz_(municipality)	2	11	1065
http://en.wikipedia.org/wiki/La_Paz_Municipality,_Bolivia	2	57	1099
http://en.wikipedia.org/wiki/Bolivia	8	3225	7469
http://en.wikipedia.org/wiki/Quechua_language	8	1878	3657
http://en.wikipedia.org/wiki/Roman_Catholic_Church	7	3284	7253
http://en.wikipedia.org/wiki/Spanish_conquest	8	1009	3816
http://en.wikipedia.org/wiki/Christopher_Columbus	7	3642	4859
http://en.wikipedia.org/wiki/Alaska	8	3523	4001
http://en.wikipedia.org/wiki/Spanish-American_War	8	3327	3673

8.3. Code

wrs.web.dal

DatabaseHelpers.java

```
package wrs.web.dal;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import wrs.web.dal.tables.RatingTable;
import wrs.web.dal.tables.WeboftrustTable;
import wrs.web.helpers.logger;
import wrs.web.rating.Rating;
import wrs.web.rating.SessionRatingDB;
import wrs.web.trust.WoT;

/*
 * @author mihai.mihaila
 */
```

```
public class DatabaseHelpers {

    public static WoT
CreateWOTFromWeboftrustTable(WeboftrustTable table) {
        try {

logger.Instance.MethodCall("CreateWOTFromWeboftrustTable");
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(table.getTrust());
            ObjectInputStream objectInputStream = new
ObjectInputStream(byteArrayInputStream);
            WoT wot = (WoT) objectInputStream.readObject();
            return wot;
        } catch (Exception exc) {
            exc.printStackTrace();
            return null;
        }
    }

    public static SessionRatingDB
GetExtractedRatings(ArrayList<RatingTable> ratingList) {
        logger.Instance.MethodCall("ExtractRatings");
        SessionRatingDB sessionRatingDB = new
SessionRatingDB();

        for (RatingTable ratingTable : ratingList) {
            Rating
rating=DatabaseHelpers.GetRatingFromRatingTable(ratingTable);
            sessionRatingDB.push(rating);
        }

        return sessionRatingDB;
    }

    public static Rating GetRatingFromRatingTable(RatingTable
ratingTable){
        Rating rating = new Rating(
                    ratingTable.getUserId().getId().toString(),
                    ratingTable.getRating(),
                    ratingTable.getCategoryId().getId(),
                    "",
                    ratingTable.getPageUrl().toString());

        return rating;
    }

    public static byte[] GetWoTBytes(WoT wot) {
        try {
            ByteArrayOutputStream byteOutputStream = new
ByteArrayOutputStream();
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(byteOutputStream);
            logger.Instance.Log("Saving WoT :" +
wot.toString());
            objectOutputStream.writeObject(wot);
            objectOutputStream.close();
            return byteOutputStream.toByteArray();
        } catch (Exception exc) {
            exc.printStackTrace();
            return new byte[0];
        }
    }
}
```

DatabaseInteraction.java

```
package wrs.web.dal;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.PersistenceUnit;
import wrs.web.dal.tables.CategoryTable;
import wrs.web.dal.tables.RatingTable;
import wrs.web.dal.tables.UserTable;
import wrs.web.dal.tables.WeboftrustTable;
import wrs.web.helpers.logger;
import wrs.web.helpers.security;
import wrs.web.trust.WoT;

/*
 * @author mihai.mihaila
 */
```

```
public class DatabaseInteraction {

    public static final int wotVersion = 1;
    @PersistenceUnit
    static EntityManagerFactory entityManagerFactory;

    public static List GetAllCategoryTables() {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        List values =
entityManager.createNamedQuery("CategoryTable.findAll").getResu
ltList();
        return values;
    }

    public static List GetAllRatingTables() {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        List values =
entityManager.createNamedQuery("RatingTable.findAll").getResult
List();
        return values;
    }

    public static List GetAllUserTables() {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        List values =
entityManager.createNamedQuery("UserTable.findAll").getResultLi
st();
        return values;
    }

    public static List GetAllWeboftrustTables() {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
```

```
        List values =
entityManager.createNamedQuery("WeboftrustTable.findAll").getRe
sultList();

        ArrayList<WeboftrustTable> returnedValues = new
ArrayList<WeboftrustTable>();
        for (Object item : values) {
            WeboftrustTable wotTable = (WeboftrustTable) item;
            if (wotTable.getVersion() ==
DatabaseInteraction.wotVersion) {
                returnedValues.add(wotTable);
            }
        }
        return returnedValues;
    }

    public static ArrayList<RatingTable>
GetRatingsByPageUrl(String pageUrl) {
        List values = DatabaseInteraction.GetAllRatingTables();
        ArrayList<RatingTable> returnedValues = new
ArrayList<RatingTable>();

        for (Object item : values) {
            RatingTable itemRatingTable = (RatingTable) item;
            if (itemRatingTable.getPageUrl().equals(pageUrl)) {
                returnedValues.add(itemRatingTable);
            }
        }

        return returnedValues;
    }

    public static UserTable GetUserTableByUsername(String
username) {
        List values = GetAllUserTables();
        UserTable returnedValue = null;
        for (Object item : values) {
            UserTable userTable = (UserTable) item;
            if (userTable.getUsername().equals(username)) {
                returnedValue = userTable;
            }
        }

        return returnedValue;
    }

    public static CategoryTable GetCategoryById(int id) {
        List values = GetAllCategoryTables();
        CategoryTable returnedValue = null;
        for (Object item : values) {
            CategoryTable categoryTable = (CategoryTable) item;
            if (categoryTable.getId() == id) {
                returnedValue = categoryTable;
            }
        }

        return returnedValue;
    }

    public static RatingTable GetRatingById(int id) {
        List values = GetAllRatingTables();
        RatingTable returnedValue = null;
        for (Object item : values) {
            RatingTable ratingTable = (RatingTable) item;
            if (ratingTable.getId() == id) {
                returnedValue = ratingTable;
            }
        }

        return returnedValue;
    }

    public static CategoryTable GetCategoryByName(String name)
{
        List values = GetAllCategoryTables();
        CategoryTable returnedValue = null;
        for (Object item : values) {
            CategoryTable categoryTable = (CategoryTable) item;
            if (categoryTable.getName().equals(name)) {
                returnedValue = categoryTable;
            }
        }

        return returnedValue;
    }

    public static UserTable GetUserTableByUserId(int userId) {
        List values = GetAllUserTables();
        UserTable returnedValue = null;
        for (Object item : values) {
            UserTable userTable = (UserTable) item;
            if (userTable.getId() == userId) {
                returnedValue = userTable;
            }
        }

        return returnedValue;
    }

    public static WeboftrustTable
GetWeboftrustTableByUsername(String username) {
        List values =
DatabaseInteraction.GetAllWeboftrustTables();
        UserTable userTable =
DatabaseInteraction.GetUserTableByUsername(username);
        WeboftrustTable returnedValue = null;
        for (Object item : values) {
            WeboftrustTable weboftrustTable = (WeboftrustTable)
item;
            if (weboftrustTable.getUserId().getId() ==
userTable.getId()) {
                returnedValue = weboftrustTable;
                break;
            }
        }

        return returnedValue;
    }

    public static WeboftrustTable
GetWeboftrustTableByUserId(int userId) {
        List values =
DatabaseInteraction.GetAllWeboftrustTables();
        UserTable userTable =
DatabaseInteraction.GetUserTableByUserId(userId);
        WeboftrustTable returnedValue = null;
        for (Object item : values) {
            WeboftrustTable weboftrustTable = (WeboftrustTable)
item;
            if (weboftrustTable.getUserId().getId() ==
userTable.getId()) {
                returnedValue = weboftrustTable;
                break;
            }
        }

        return returnedValue;
    }

    public static WeboftrustTable GetWeboftrustTableById(int
id) {
        List values =
DatabaseInteraction.GetAllWeboftrustTables();
        WeboftrustTable returnedValue = null;
        for (Object item : values) {
            WeboftrustTable weboftrustTable = (WeboftrustTable)
item;
            if (weboftrustTable.getId() == id) {
                returnedValue = weboftrustTable;
                break;
            }
        }

        return returnedValue;
    }

    public static WeboftrustTable
CreateEmptyWeboftrustTable(String username) {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        EntityTransaction userTransaction =
entityManager.getTransaction();

        userTransaction.begin();

        WeboftrustTable wotTable = new WeboftrustTable();

        UserTable userTable = GetUserTableByUsername(username);
        WoT wot = new WoT(userTable.getId().toString());

        wotTable.setUserId(userTable);
        wotTable.setTrust(DatabaseHelpers.GetWoTBytes(wot));
        wotTable.setVersion(DatabaseInteraction.wotVersion);
        wotTable.setId(-1);
        entityManager.persist(wotTable);
        userTransaction.commit();
        return wotTable;
    }

    public static RatingTable AddRating(String pageUrl, int
rating, int categoryId, boolean experience, String username) {
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        EntityTransaction userTransaction =
entityManager.getTransaction();

        userTransaction.begin();

        RatingTable ratingTable = new RatingTable();

        UserTable userTable = GetUserTableByUsername(username);
        CategoryTable categoryTable =
GetCategoryById(categoryId);

        ratingTable.setUserId(userTable);
        ratingTable.setCategoryId(categoryTable);
        ratingTable.setRating(rating);
        ratingTable.setPageUrl(pageUrl);
        ratingTable.setId(-1);

        Date date = new Date();

        ratingTable.setRatingTime(date);

        entityManager.persist(ratingTable);
        userTransaction.commit();
        return ratingTable;
    }

    public static UserTable CreateUser(String username, String
password) throws Exception {
        UserTable alreadyExistingUsername =
GetUserTableByUsername(username);
        if (alreadyExistingUsername != null) {
            throw new Exception("Already existing user. Please
pick another username");
        }
        DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        EntityTransaction userTransaction =
entityManager.getTransaction();

        userTransaction.begin();

        UserTable userTable = new UserTable();
        userTable.setUsername(username);
        userTable.setPasswordHash(security.GetHash(password));
        userTable.setId(-1);

        entityManager.persist(userTable);
        userTransaction.commit();
        return userTable;
    }

    public static void WriteWOTToDatabase(WoT wot, int wotId) {
        try {
            logger.Instance.MethodCall("WriteWOTToDatabase");

            DatabaseInteraction.entityManagerFactory =
Persistence.createEntityManagerFactory("WRSPersistanceUnit");
            EntityManager entityManager =
entityManagerFactory.createEntityManager();
            entityManager.getTransaction().begin();

            WeboftrustTable wotTable =
entityManager.find(WeboftrustTable.class, wotId);

wotTable.setTrust(DatabaseHelpers.GetWoTBytes(wot));
            entityManager.persist(wotTable);
            entityManager.getTransaction().commit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

wrs.web.external

WikiTrustResponseBuilder.java

```
package wrs.web.external;

import java.text.DecimalFormat;
import wrs.web.dal.*;
import wrs.web.dal.tables.RatingTable;
import wrs.web.dal.tables.WeboftrustTable;
import wrs.web.helpers.logger;
import wrs.web.helpers.wikitrust;
import wrs.web.rating.*;
import wrs.web.resources.*;
import wrs.web.trust.*;

/*
 * @author mihai.mihaila
 */
public class WikiTrustResponseBuilder implements
IResponseBuilder {

    public static final String WikiTrustName = "WikiTrust";
    private boolean initialized = false;
    private TrustUpdater trustUpdater;
    private WoT webOfTrust = null;
    private int wotId = -1;
    private String userId = "";
    private String password = "";
    private RatingCalculator ratingCalculator = null;
    private Rating wikiTrustRating = null;

    public RatingResponse GetRating(String pageUrl, String
username) {
        this.InitializeVariables(username);
        this.UpdateTrust(pageUrl, true, false);
        // return normal trust value
        InteractionData computedAverage =
this.ratingCalculator.computeAverageInteractionData(17);
        RatingResponse ratingResponse =
this.GetCategoryAndRating(computedAverage);

        if (isRatingNull(ratingResponse)) {
            this.GetWikitrustRating(pageUrl);
            // return wikitrust rating
            ratingResponse = new RatingResponse();
            if (wikiTrustRating != null) {
                if (wikiTrustRating.getCategory() <= 15) {
                    ratingResponse.category =
wikiTrustRating.getCategory() + "";
                } else {
                    ratingResponse.category = "-1";
                }
                ratingResponse.categoryRatingPercentage = "";
                ratingResponse.rating =
wikiTrustRating.getRating() + "";
            } else {
                ratingResponse.category = "Not assigned";
                ratingResponse.categoryRatingPercentage = "";
                ratingResponse.rating = "Unknown";
            }
        }
        return ratingResponse;
    }

    private boolean isRatingNull(RatingResponse rating) {
        String nonExistingPrefix = "Not enough";
        if (rating.rating.startsWith(nonExistingPrefix) &&
rating.categoryRatingPercentage.startsWith(nonExistingPrefix))
{
            return true;
        }

        return false;
    }

    public GenericResponse SetRating(String pageUrl, int
rating, int categoryId, boolean experience, String username) {
        //add rating to database
        RatingTable ratingTable =
DatabaseInteraction.AddRating(pageUrl, rating, categoryId,
experience, username);
        this.InitializeVariables(username);
        // modify wikitrust category
        this.ModifyWikitrustRating(pageUrl, categoryId);
        this.UpdateTrust(pageUrl, false, true);
        this.trustUpdater.updateAndInsertRatingsToWoT(rating,
categoryId, experience);
        return new GenericResponse(true);
    }

    public int GetImplementationId() {
        return 1;
    }

    private RatingResponse GetCategoryAndRating(InteractionData
interactionData) {
        RatingResponse ratingResponse = new RatingResponse();
        try {
            DecimalFormat decimalFormat = new
DecimalFormat("#0.00");
            if
(String.valueOf(interactionData.getAverageRating()).contains("N
aN")) {
                ratingResponse.rating = "Not enough information
for a rating";
                logger.Instance.Log("Not enough information for
a rating");
            } else {
                ratingResponse.rating =
decimalFormat.format(interactionData.getAverageRating());
                logger.Instance.Log("Rating:" +
decimalFormat.format(interactionData.getAverageRating()));
            }

            if (interactionData.getCategoryType() != -1) {
                ratingResponse.categoryRatingPercentage =
decimalFormat.format(interactionData.getCategoryAverage() *
100);
                ratingResponse.category =
interactionData.getCategoryType() + "";
                logger.Instance.Log("Category:" +
decimalFormat.format(interactionData.getCategoryAverage() *
100) + " % " + interactionData.getCategoryType());
            } else {
                ratingResponse.categoryRatingPercentage = "Not
enough information for a category";
                ratingResponse.category = "-1";
                logger.Instance.Log("Not enough information for
a category");
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return ratingResponse;
    }

    private void InitializeVariables(String username) {
        logger.Instance.MethodCall("InitializeVariables");
        if (this.initialized) {
            return;
        } else {
            this.initialized = true;
        }

        WeboftrustTable wotTable =
DatabaseInteraction.GetWeboftrustTableByUsername(username);
        if (wotTable == null) {
            wotTable =
DatabaseInteraction.CreateEmptyWeboftrustTable(username);
            wotTable =
DatabaseInteraction.GetWeboftrustTableByUsername(username);
        }

        this.wotId = wotTable.getId();
        this.webOfTrust =
DatabaseHelpers.CreateWOTFromWeboftrustTable(wotTable);
        this.userId =
DatabaseInteraction.GetUserTableByUsername(username).getId().to
String();
        logger.Instance.Log("WoT from database:" +
this.webOfTrust.toString());
    }

    private void UpdateTrust(String pageUrl, boolean
updateCurrentRating, boolean prepareForInsertAverage) {
        boolean hasRating = true;
        SessionRatingDB sessionRatingDB =
DatabaseHelpers.GetExtractedRatings(DatabaseInteraction.GetRati
ngsByPageUrl(pageUrl));

        sessionRatingDB.prettyPrintObject();
        if (false) {
            //if (updateCurrentRating) {
            // If the owner of the WoT is found to have given a
rating, that rating is retrieved and removed from the
SessionRatingDB.
            Rating ratingByCurrentUser =
sessionRatingDB.checkHasOwner(this.userId);

            if (ratingByCurrentUser != null) {
                logger.Instance.Log("User has rated this page
before, checking for helpers ratings.");
                InteractionData interaction =
this.webOfTrust.getInteraction(ratingByCurrentUser.getArticleUR
L());
                if (interaction != null) {
                    this.trustUpdater = new
TrustUpdater(sessionRatingDB, interaction, this.webOfTrust,
this.wotId);

this.trustUpdater.updateAndInsertRatingsToWoT(ratingByCurrentUs
er.getRating(), ratingByCurrentUser.getCategory(),
interaction.getExperience() == 1);
                }
            }
        }

        // An rating is calculated based on the trust values
from the WoT
        this.ratingCalculator = new
RatingCalculator(sessionRatingDB, this.webOfTrust);

        if (prepareForInsertAverage) {
            InteractionData computedAverage =
ratingCalculator.computeAverageInteractionData(17);
            // Average is inserted along with the HTML to cast
own vote and  feedback
            this.trustUpdater = new
TrustUpdater(sessionRatingDB, computedAverage, this.webOfTrust,
this.wotId);
        }
    }

    public void GetWikitrustRating(String pageUrl) {
        Rating previousWikitrustRating =
webOfTrust.getRatingOfUserForPage(WikiTrustName, pageUrl);
        if (previousWikitrustRating != null) {
            wikiTrustRating = previousWikitrustRating;
        } else {
            int wikitrustRatingValue =
wikitrust.getPageTrust(pageUrl);
            if (wikitrustRatingValue != -1) {
                wikiTrustRating = new Rating(WikiTrustName,
wikitrustRatingValue, 16, "", pageUrl);

                Reviewer wikitrustReviewer = new Reviewer(17);
                wikitrustReviewer.setUsername(WikiTrustName);

wikitrustReviewer.insertRating(wikiTrustRating);
                webOfTrust.insertReviewer(wikitrustReviewer);

DatabaseInteraction.WriteWOTToDatabase(webOfTrust, wotId);
            }
        }
    }

    private void ModifyWikitrustRating(String pageUrl, int
categoryId) {
        Rating previousWikitrustRating =
webOfTrust.getRatingOfUserForPage(WikiTrustName, pageUrl);
        if (previousWikitrustRating != null) {
            previousWikitrustRating.setCategory(categoryId);

webOfTrust.insertNewRatingOfUserForPage(WikiTrustName, pageUrl,
previousWikitrustRating);
        }
    }
}
```

wrs.web.helpers

security.java

```
package wrs.web.helpers;

import java.security.MessageDigest;
```

```
/**
 *
 * @author mihai.mihaila
 */
public class security {

    public static String GetHash(String message) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-
256");
            md.update(message.getBytes());
            byte[] hashBytes = md.digest();

            StringBuffer stringBuffer = new StringBuffer();
            for (int i = 0; i < hashBytes.length; i++) {

stringBuffer.append(Integer.toString((hashBytes[i] & 0xff) +
0x100, 16).substring(1));
            }

            return stringBuffer.toString();

        } catch (Exception exc) {
        }

        return null;
    }
}
```

wikipedia.java

```
package wrs.web.helpers;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.JsonToken;
```

```
/**
 *
 * @author mihai.mihaila
 */
public class wikipedia {

    public static final String wikipediaAddress =
"http://en.wikipedia.org/wiki/";

    public static ArrayList<Integer> GetLatestRevisions(String
pageUrl, int revisionsCount) {

        ArrayList<Integer> pageRevisions = new
ArrayList<Integer>();

        String articleName = "";

        int startIndex = wikipedia.wikipediaAddress.length();
        int endIndex = pageUrl.length();

        logger.Instance.Log("Extracting page name from
address:" + pageUrl + " startIndex:" + startIndex + "
endIndex:" + endIndex);;
        articleName = pageUrl.substring(startIndex, endIndex);

        articleName = GetSanitizedArticleName(articleName);

        String utfArticleName = articleName;
        try {
            utfArticleName = URLEncoder.encode(articleName,
"UTF-8");
        } catch (Exception exc) {
            exc.printStackTrace();
            //ignore the error
        }

        String address =
"http://en.wikipedia.org/w/api.php?action=query&format=json&pro
p=revisions&rvprop=ids&rvlimit=" + revisionsCount + "&titles="
+ utfArticleName;

        try {
            URL url = new URL(address);
            HttpURLConnection httpUrlConnection =
(HttpURLConnection) url.openConnection();
            httpUrlConnection.connect();
            InputStreamReader inputStreamReader = new
InputStreamReader(httpUrlConnection.getInputStream());
            BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);
            StringBuilder stringBuilder = new StringBuilder();
            String line = "";
            while ((line = bufferedReader.readLine()) != null)
{
                stringBuilder.append(line);
            }

            bufferedReader.close();
            inputStreamReader.close();
            httpUrlConnection.disconnect();

            JsonFactory factory = new JsonFactory();
            JsonParser parser =
factory.createJsonParser(stringBuilder.toString());

            JsonToken token = null;
            while ((token = parser.nextToken()) != null) {
                if (parser.getCurrentName() != null &&
parser.getCurrentName().equals("revid")) {
                    token = parser.nextToken();

pageRevisions.add(Integer.parseInt(parser.getText()));
                }
            }
        } catch (Exception exc) {
            exc.printStackTrace();;
        }

        return pageRevisions;
    }

    //get the right article name, dealing with underscores and
page redirects
    public static String GetSanitizedArticleName(String
articleName) {
        String returnedArticleName = articleName;
        String utfArticleName = articleName;
        try {
            utfArticleName = URLEncoder.encode(articleName,
"UTF-8");
        } catch (Exception exc) {
            exc.printStackTrace();
            //ignore the error
        }

        String address =
"http://en.wikipedia.org/w/api.php?action=query&format=json&red
irects&titles=" + utfArticleName;

        try {
            URL url = new URL(address);
            HttpURLConnection httpUrlConnection =
(HttpURLConnection) url.openConnection();
            httpUrlConnection.connect();
            InputStreamReader inputStreamReader = new
InputStreamReader(httpUrlConnection.getInputStream());
            BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);
            StringBuilder stringBuilder = new StringBuilder();
            String line = "";
            while ((line = bufferedReader.readLine()) != null)
{
                stringBuilder.append(line);
            }

            bufferedReader.close();
            inputStreamReader.close();
            httpUrlConnection.disconnect();

            JsonFactory factory = new JsonFactory();
            JsonParser parser =
factory.createJsonParser(stringBuilder.toString());

            JsonToken token = null;
            while ((token = parser.nextToken()) != null) {
                if (parser.getCurrentName() != null &&
parser.getCurrentName().equals("redirects")) {
                    returnedArticleName =
GetNextToValue(parser, token);
                    break;
                }
            }
        } catch (Exception exc) {
            exc.printStackTrace();;
        }

        return returnedArticleName;
    }

    private static String GetNextToValue(JsonParser parser,
JsonToken token) throws IOException {
        while ((token = parser.nextToken()) != null) {
            if (parser.getCurrentName() != null &&
parser.getCurrentName().equals("to")) {
                token = parser.nextToken();
                return parser.getText();
            }
        }

        return "";
    }

    //get the right article name, dealing with underscores and
page redirects
    public static int GetPageId(String pageUrl) {

        String articleName = "";

        int startIndex = wikipedia.wikipediaAddress.length();
        int endIndex = pageUrl.length();

        logger.Instance.Log("Retrieving page id for article:" +
pageUrl + " startIndex:" + startIndex + " endIndex:" +
endIndex);;
        articleName = pageUrl.substring(startIndex, endIndex);

        String utfArticleName = articleName;
        try {
            utfArticleName = URLEncoder.encode(articleName,
"UTF-8");
        } catch (Exception exc) {
            exc.printStackTrace();
        }
        String address =
"http://en.wikipedia.org/w/api.php?action=query&format=json&red
irects&titles=" + utfArticleName;

        try {
            URL url = new URL(address);
            HttpURLConnection httpUrlConnection =
(HttpURLConnection) url.openConnection();
            httpUrlConnection.connect();
            InputStreamReader inputStreamReader = new
InputStreamReader(httpUrlConnection.getInputStream());
            BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);
            StringBuilder stringBuilder = new StringBuilder();
            String line = "";
            while ((line = bufferedReader.readLine()) != null)
{
                stringBuilder.append(line);
            }

            bufferedReader.close();
            inputStreamReader.close();
            httpUrlConnection.disconnect();

            JsonFactory factory = new JsonFactory();
            JsonParser parser =
factory.createJsonParser(stringBuilder.toString());

            JsonToken token = null;
            while ((token = parser.nextToken()) != null) {
                if (parser.getCurrentName() != null &&
parser.getCurrentName().equals("pageid")) {
                    token = parser.nextToken();
                    return Integer.parseInt(parser.getText());
                }
            }
        } catch (Exception exc) {
            exc.printStackTrace();;
        }

        return -1;
    }
}
```

wikitrust.java

```
package wrs.web.helpers;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;

/**
 *
 * @author mihai.mihaila
 */
public class wikitrust {

    public static int getPageTrust(String articleUrl) {
        int rating = 0;

        long start = System.currentTimeMillis();
        long end1 = 0;
        long end2 = 0;
        long end3 = 0;

        ArrayList<Integer> revisionIds =
wikipedia.GetLatestRevisions(articleUrl, 1);

        int pageId = wikipedia.GetPageId(articleUrl);
        String pageUrl =
"http://en.collaborativetrust.com/WikiTrust/RemoteAPI?method=wi
kimarkup&pageid=" + pageId + "&revid=" + revisionIds.get(0);
        ArrayList<Integer> trustArray = new
ArrayList<Integer>();
        ArrayList<Integer> trustWeightArray = new
ArrayList<Integer>();
        try {
            URL url = new URL(pageUrl);
            HttpURLConnection httpUrlConnection =
(HttpURLConnection) url.openConnection();
            httpUrlConnection.connect();
            InputStreamReader inputStreamReader = new
InputStreamReader(httpUrlConnection.getInputStream());
            BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);
```

```
            StringBuilder stringBuilder = new StringBuilder();
            String line = "";
            while ((line = bufferedReader.readLine()) != null)
{
                stringBuilder.append(line);
            }

            bufferedReader.close();
            inputStreamReader.close();
            httpUrlConnection.disconnect();

            end1 = System.currentTimeMillis();

            int index = stringBuilder.indexOf("{{#t:", 0);
            int previousClosingBracket = -1;

            while (index >= 0) {
                int commaIndex = stringBuilder.indexOf(",",
index);
                int closingBracket =
stringBuilder.indexOf("}}", index) + 2;

                String trust = stringBuilder.substring(index +
5, commaIndex);
                trustArray.add(Integer.parseInt(trust));


                if (previousClosingBracket != -1) {
                    trustWeightArray.add(index -
previousClosingBracket);
                }

                previousClosingBracket = closingBracket;
                index = stringBuilder.indexOf("{{#t:",
commaIndex);
            }

            trustWeightArray.add(stringBuilder.length() -
previousClosingBracket);
            end2 = System.currentTimeMillis();

        } catch (Exception exc) {
            exc.printStackTrace();;
        }
        int totalWeight = 0;

        for (int i = 0; i < trustArray.size(); i++) {
            rating += trustArray.get(i) *
trustWeightArray.get(i);
            totalWeight += trustWeightArray.get(i);
        }

        rating = rating / totalWeight;
        rating = (rating * 9) / 11;
        rating+=1;


        end3 = System.currentTimeMillis();

        System.out.println("v5." + "      " + articleUrl + "
      " + rating + "    " + trustArray.size() + "     " + (end1
- start) + "      " + (end2 - end1) + "   " + (end3 - end2) + "
      " + (end3 - start));

        return rating;
    }
}
```

wrs.web.internals

WRSInternalResponses.java

```
package wrs.web.internals;

import java.util.List;
import wrs.web.dal.DatabaseInteraction;
import wrs.web.dal.tables.CategoryTable;
import wrs.web.resources.CategoryItemResponse;
import wrs.web.resources.CategoryResponse;

/**
 *
 * @author mihai.mihaila
 */
public class WRSInternalResponses {
    public static CategoryResponse GetCategories() {
        CategoryResponse bean = new CategoryResponse();
        List categories =
DatabaseInteraction.GetAllCategoryTables();

        for (Object categoryObject : categories) {
            CategoryTable categoryTable = (CategoryTable)
categoryObject;
            bean.Items.add(new
CategoryItemResponse(categoryTable.getId(),
categoryTable.getName()));
        }

        return bean;
    }
}
```

wrs.web.rating

InteractionData.java

```
package wrs.web.rating;

import java.io.Serializable;
import wrs.web.helpers.logger;

/*
 * User: mihai.mihaila
 * Date: 5/13/11
 * Time: 9:59 PM
 */
public class InteractionData implements Serializable
{
    /*
    * 0 = Rating average,
    * 1 = Category Average,
    * 2 = Category Type,
    * 3 = Experience (1 good, 0 bad)
    */

    // fields

    private double averageRating;
    private double categoryAverage;
    private int categoryType;
    private int experience;
    // getters and setters
    public int getCategoryType()
    {
        return categoryType;
    }

    public void setCategoryType(int categoryType)
    {
        this.categoryType = categoryType;
    }

    public double getCategoryAverage()
    {
        return categoryAverage;
    }

    public void setCategoryAverage(double categoryAverage)
    {
        this.categoryAverage = categoryAverage;
    }

    public double getAverageRating()
    {
        return averageRating;
    }

    public void setAverageRating(double averageRating)
    {
        this.averageRating = averageRating;
    }

    public int getExperience()
    {
        return experience;
    }

    public void setExperience(int experience)
    {
        this.experience = experience;
    }
    public InteractionData()
    {
        this.averageRating = 0;
        this.categoryAverage = 0;
        this.categoryType = -1;
        this.experience = -1;
    }

    public InteractionData(InteractionData other)
    {
        this.setAverageRating(other.getAverageRating());
        this.setExperience(other.getExperience());
        this.setCategoryAverage(other.getCategoryAverage());
        this.setCategoryType(other.getCategoryType());
    }

    @Override
    public String toString()
    {
        String returnValue = "";
        returnValue+="Rating:"+this.getAverageRating()+
logger.newline;
        returnValue+="Category
Average:"+this.getCategoryAverage()+ logger.newline;
        returnValue+="Category Type:"+this.getCategoryType()+
logger.newline;
        returnValue+="Experience:"+this.getExperience()+
logger.newline;
        return returnValue;
    }
}
```

wrs.web.resources

CategoryItemResponse.java

```
    package wrs.web.resources;

    import javax.xml.bind.annotation.XmlRootElement;

    /**
     *
     * @author mihai.mihaila
     */

    @XmlRootElement
    public class CategoryItemResponse {
         public int id;
        public String name;

        public CategoryItemResponse() {
        }

        public CategoryItemResponse(int id, String name) {
            this.id = id;
            this.name = name;
        }
    }
```

CategoryResponse.java

```
package wrs.web.resources;

import java.util.ArrayList;
import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author mihai.mihaila
 */
@XmlRootElement
public class CategoryResponse extends GenericResponse{
    public ArrayList<CategoryItemResponse> Items;
    public CategoryResponse() {
        this.Items=new ArrayList<CategoryItemResponse>();
    }
}
```

GenericResponse.java

```
package wrs.web.resources;

import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author mihai.mihaila
 */
@XmlRootElement
public class GenericResponse {

    public String result = "success";
    public String exception;

    public GenericResponse() {
    }

    public GenericResponse(String exception) {
        this.exception = exception;
    }

    public GenericResponse(boolean result) {
        if (result) {
            this.result = "success";
        } else {
            this.result = "fail";
        }
    }

    public GenericResponse(boolean result, String exception) {
        if (result) {
            this.result = "success";
        } else {
            this.result = "fail";
        }

        this.exception = exception;
    }
}
```

IResponseBuilder.java

```
package wrs.web.resources;

/**
 *
 * @author mihai.mihaila
 */
public interface IResponseBuilder {

    public int GetImplementationId();
    public RatingResponse GetRating(String pageUrl,String
username);

    public GenericResponse SetRating(String pageUrl,int
rating,int categoryId,boolean experience,String username);
}
```

RatingResponse.java

```
package wrs.web.resources;

import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author mihai.mihaila
 */
@XmlRootElement
public class RatingResponse extends GenericResponse{

    public String rating;

    public String categoryRatingPercentage;

    public String category;

    public RatingResponse()
    {

    }
}
```

WRSResource.java

```
package wrs.web.resources;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Enumeration;
import wrs.web.external.*;
import java.util.List;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import wrs.web.dal.DatabaseInteraction;
import wrs.web.dal.IncorrectParameterValueException;
import wrs.web.dal.tables.UserTable;
import wrs.web.helpers.security;
import wrs.web.internals.WRSInternalResponses;

@Path("/wrs/")
public class WRSResource {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public GenericResponse processRequest(
            @DefaultValue("") @QueryParam("method") String
method,
            @DefaultValue("1") @QueryParam("id") String id,
            @DefaultValue("1") @QueryParam("impl") int
implementation,
            @DefaultValue("") @QueryParam("pageUrl") String
pageUrl,
            @DefaultValue("") @QueryParam("username") String
username,
            @DefaultValue("") @QueryParam("password") String
password,
            @DefaultValue("0") @QueryParam("rating") int
rating,
            @DefaultValue("0") @QueryParam("categoryRating")
int categoryRating,
            @DefaultValue("") @QueryParam("experience") boolean
experience) {

        System.out.println("Welcome");

        if (method.equals("")) {
            return new GenericResponse("Method not found");
        }

        if (method.equals("createUser")) {
            if (username.compareTo("") != 0 &&
password.compareTo("") != 0) {
                try{
                UserTable userTable =
DatabaseInteraction.CreateUser(username, password);
                if (userTable.getId() > 0) {
                    return new GenericResponse(true);
                }
                }
                catch(Exception exc)
                {
                    return new
GenericResponse(false,exc.getMessage());
                }
            }

            return new GenericResponse(false);
        }

        if (method.equals("login")) {
            try {
                this.login(username, password);
                return new GenericResponse(true);
            } catch (IncorrectParameterValueException exc) {
                return new GenericResponse(exc.getMessage());
            }
        }

        if (method.equals("getCategories")) {

            return WRSInternalResponses.GetCategories();
        }

        if (method.equals("getRating")) {
            IResponseBuilder responseBuilder = null;
            try {
                responseBuilder =
this.GetRequestedResponseBuilder(implementation);
                this.login(username, password);

            } catch (IncorrectParameterValueException exc) {
                return new
GenericResponse(false,exc.getMessage());
            } catch (DuplicateImplementationException exc) {
                return new
GenericResponse(false,exc.getMessage());
            }

            if (responseBuilder == null) {
                return new GenericResponse("No implementations
found");
            }

            RatingResponse response =
responseBuilder.GetRating(pageUrl, username);
            return response;
        }

        if (method.equals("setRating")) {

            IResponseBuilder responseBuilder = null;
            try {
                responseBuilder =
this.GetRequestedResponseBuilder(implementation);
                this.login(username, password);

            } catch (IncorrectParameterValueException exc) {
                return new
GenericResponse(false,exc.getMessage());
            } catch (DuplicateImplementationException exc) {
                return new
GenericResponse(false,exc.getMessage());
            }

            if (responseBuilder == null) {
                return new GenericResponse("No implementations
found");
            }

            return responseBuilder.SetRating(pageUrl, rating,
categoryRating, experience, username);
        }

        return new GenericResponse("Invalid parameters");
    }

    public void login(String username, String password) throws
IncorrectParameterValueException {
        UserTable userTable =
DatabaseInteraction.GetUserTableByUsername(username);
        if (userTable == null) {
            throw new
IncorrectParameterValueException("Incorrect username. Please
try again!");
        }

        if
(userTable.getPasswordHash().compareTo(security.GetHash(passwor
d)) != 0) {
            throw new
IncorrectParameterValueException("Incorrect password. Please
try again!");
        }
    }

    public IResponseBuilder GetRequestedResponseBuilder(int
implementationId) throws DuplicateImplementationException {
        IResponseBuilder responseBuilder = null;
        ArrayList<IResponseBuilder> responseBuilderArray = new
ArrayList<IResponseBuilder>();
        responseBuilderArray.add(new
WikiTrustResponseBuilder());

        try {
            Class[] classes =
this.loadClassesFromExternalPackage();
            responseBuilderArray =
this.loadFromPlugin(classes);
        } catch (Exception exc) {
        }

        IResponseBuilder plugin = null;
        if (plugin != null) {
            responseBuilderArray.add(plugin);
        }

        for (IResponseBuilder responseBuilderItem :
responseBuilderArray) {
            if (responseBuilderItem.GetImplementationId() ==
implementationId) {
                if (responseBuilder != null) {
                    throw new
DuplicateImplementationException("Duplicate implementations
found");
                } else {
                    responseBuilder = responseBuilderItem;
                }
            }
        }

        return responseBuilder;
    }

    public Class[] loadClassesFromExternalPackage() throws
IOException, ClassNotFoundException {
        ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
        assert classLoader != null;
        String packageName = "wrs.web.external";
        String path = packageName.replace('.', '/');
        Enumeration<URL> resources =
classLoader.getResources(path);
        List<File> dirs = new ArrayList<File>();
        while (resources.hasMoreElements()) {
            URL resource = resources.nextElement();
            dirs.add(new File(resource.getFile()));
        }
        ArrayList<Class> classes = new ArrayList<Class>();
        for (File directory : dirs) {
            classes.addAll(findClasses(directory,
packageName));
        }

        return classes.toArray(new Class[classes.size()]);
    }

    private static List<Class> findClasses(File directory,
String packageName) throws ClassNotFoundException {
        List<Class> classes = new ArrayList<Class>();
        if (!directory.exists()) {
            return classes;
        }

        File[] files = directory.listFiles();
        for (File file : files) {
            if (file.isDirectory()) {
                assert !file.getName().contains(".");
                classes.addAll(findClasses(file, packageName +
"." + file.getName()));
            } else if (file.getName().endsWith(".class")) {
                classes.add(Class.forName(packageName + '.' +
file.getName().substring(0, file.getName().length() - 6)));
            }
        }
        return classes;
    }

    private ArrayList<IResponseBuilder> loadFromPlugin(Class[]
classes) {
        ArrayList<IResponseBuilder> loadedClasses = new
ArrayList<IResponseBuilder>();
        for (Class loadedClass : classes) {
            try {
                Object obj = loadedClass.newInstance();
                if (obj instanceof IResponseBuilder) {
                    loadedClasses.add((IResponseBuilder) obj);
                }
            } catch (Exception exc) {
            }
        }
        return loadedClasses;
    }
}
```

WRS Chrome Extension

global.js

```
var notifications = [];
var categories = {};

var wrsUsernameString = "wrsUsername";
var wrsPasswordString = "wrsPassword";
var displayNotificationsString = "displayNotifications";

//local machine
//var secureServerName = "https://Mihai-Acer-PC:8181";
```

```
//var secureServerName = "http://Mihai-Acer-PC:8080";
//var normalServerName = "http://Mihai-Acer-PC:8080";


var secureServerName = "http://vmwrs.imm.dtu.dk:8080";
var normalServerName = "http://vmwrs.imm.dtu.dk:8080";

var reqGetRating = new XMLHttpRequest();
var reqCategories = new XMLHttpRequest();
var waitingTimeout = -1;

function showNotification(title, textToDisplay) {
    if (waitingTimeout != -1) {
        clearInterval(waitingTimeout);
        closeNotificationTimerElapsed();
    }

    //chrome.browserAction.setBadgeText({text:select.value});
    var notification = webkitNotifications.createNotification(
                      'wrs icon.png',
                      title,
                      textToDisplay);

    notifications.push(notification);
    notification.show();

    waitingTimeout = setTimeout(closeNotificationTimerElapsed,
10000);
}

function closeNotificationTimerElapsed() {
    waitingTimeout = -1;
    closeAllNotification();
}

function closeAllNotification() {
    while (notifications.length > 0) {
        var notification = notifications.pop();
        notification.cancel();
    }
}

function loadRatingForPopup(username, password, pageUrl) {
    reqGetRating.open("GET", secureServerName + "/wrs-
webapp/wrs?method=getRating&username=" + username +
"&password=" + password + "&pageUrl=" + pageUrl, true);
    reqGetRating.onload = processRequestGetRating;
    reqGetRating.send();
}


function cleanAddress(address) {
    var indexOfBreak = address.indexOf("#", 0);
    if (indexOfBreak >= 0) {
        address = address.substring(0, indexOfBreak);
    }

    return address;
}

function getTitleFromUrl(url) {
    var prefix = "http://en.wikipedia.org/wiki/";
    var address = url.substring(prefix.length);
    var replaceWithSpace=["_"];
    for(var i=0;i<replaceWithSpace.length;i++)
    {
        while (address.indexOf(replaceWithSpace[i]) >= 0) {
            address = address.replace(replaceWithSpace[i], "
");
        }
    }

    return address;
}

function processRequestGetRating() {
    var ratingDiv = document.getElementById("ratingDiv");
    var categoryRatingDiv =
document.getElementById("categoryRatingDiv");

    var jsonValue = JSON.parse(reqGetRating.response);
    if (jsonValue != null) {
        if (containsException(jsonValue)) {
            var loginText =
document.getElementById("loginText");
            loginText.innerText = "You are not logged in!";
        }
        else {
            ratingDiv.innerText = jsonValue.rating;
            if (jsonValue.categoryRatingPercentage != "") {
                categoryRatingDiv.innerText =
jsonValue.categoryRatingPercentage + " % ";
            }

            if (jsonValue.category != "-1") {
                categoryRatingDiv.innerText +=
categories[jsonValue.category];
            }
            else {
                categoryRatingDiv.innerText += "Unassigned
category";
            }
        }
    }
}

function containsException(jsonValue) {
    if (jsonValue.exception != undefined) {
        return true;
    }
    else {
        return false;
    }
}

function loadRatingForBackground(username, password, pageUrl,
tabTitle) {
    if (localStorage[displayNotificationsString] == "true") {
        return;
    }
    else {
        reqGetRating.open("GET", secureServerName + "/wrs-
webapp/wrs?method=getRating&username=" + username +
"&password=" + password + "&pageUrl=" + pageUrl, true);
        reqGetRating.onload = function () {
            processRequestGetRatingBackground(tabTitle);
        }
        reqGetRating.send();
    }
}
```

```
function processRequestGetRatingBackground(tabTitle) {
    var jsonValue = JSON.parse(reqGetRating.response);
    if (jsonValue != null) {

        if (containsException(jsonValue)) {
            showNotification("WRS Rating", "You are not logged
in. Use the WRS navigation bar icon to login");
        }
        else {
            var ratingValue = jsonValue.rating;
            var categoryPercentage =
jsonValue.categoryRatingPercentage;
            var category = jsonValue.category;

            showNotification(tabTitle,
formatRatingPopupText(ratingValue, categoryPercentage,
category));
        }
    }
}

function formatRatingPopupText(ratingValue, categoryPercentage,
category) {
    var text = "Rating: " + ratingValue + "; ";
    if (categoryPercentage != "") {
        text += categoryPercentage + " % ";
    }

    if (category != "-1") {
        text += categories[category];
    }
    else {
        text += " Unassigned category";
    }

    return text;
}

function addCategoriesToPopup() {
    addOption("", -1);
    for (key in categories) {
        addOption(categories[key], key);
    }
}
```

```
function loadCategories() {
    reqCategories.open("GET", normalServerName + "/wrs-
webapp/wrs?method=getCategories", true);
    reqCategories.onload = processRequestCategories;
    reqCategories.send();
}

function loadCategoriesAndAddOptions() {
    reqCategories.open("GET", normalServerName + "/wrs-
webapp/wrs?method=getCategories", true);
    reqCategories.onload =
processRequestCategoriesAndAddOptions;
    reqCategories.send();
}

function processRequestCategoriesAndAddOptions() {
    processRequestCategories();
    addCategoriesToPopup();
}


function processRequestCategories() {
    var jsonValue = JSON.parse(reqCategories.response);
    if (jsonValue != null && jsonValue.Items != null) {
        for (var i = 0; i < jsonValue.Items.length; i++) {
            categories[jsonValue.Items[i].id] =
jsonValue.Items[i].name;
        }
    }
}
```

popupPage.html

```
<html>
<head>
    <title>Wikipedia Recommender System </title>
    <style type="text/css">
        body
        {
            min-width: 300px;
            max-height: 300px;
        }
        #categoryComboBox
        {
            width: 200px;
        }
    </style>
    <script type="text/javascript" src="../sjcl.js"></script>
    <script type="text/javascript" src="../global.js"></script>
    <script language="javascript" type="text/javascript">

        var rating = undefined;
        var experience = undefined;
        var categoryId = undefined;

        var pageUrl;
        var username = localStorage[wrsUsernameString];
        var password = localStorage[wrsPasswordString];

        var reqSetRating = new XMLHttpRequest();

        function loaded() {



            chrome.windows.getCurrent(function (window) {
                chrome.tabs.getSelected(window.id, function
(tab) {
                    pageUrl = cleanAddress(tab.url);
                    loadCategoriesAndAddOptions();
                    loadRatingForPopup(username, password,
pageUrl);
                    Initialize();
                });
            });
        }




        function NavigateToOptions() {
            window.open("options.html");
        }

        function Initialize() {
            var loginButton =
document.getElementById("loginButton");
```

```
            var divUsername =
document.getElementById("divUsername");
            var loginText =
document.getElementById("loginText");
            if (username != undefined && username !=
"undefined") {
                loginText.innerText = "logged in as ";
                divUsername.innerText = username;
                loginButton.value = "Log out";
            } else {
                loginText.innerText = "";
                divUsername.innerText = "";
                loginButton.value = "Log in";
            }
        }

        function addOption(display, value) {
            var select =
document.getElementById("categoryComboBox");
            var newOption = document.createElement("option");
            newOption.value = value;
            newOption.innerText = display;
            select.appendChild(newOption);
        }

        function callbackExperience(button) {
            if (checkCategorySelected()) {
                experience = button.value;
                disablePanel("panelExperience");
                checkAndSubmitRating();
            }
        }

        function checkCategorySelected() {
            var select =
document.getElementById("categoryComboBox");
            if (select.value == -1) {
                showMessage("Select a category first!");
                return false;
            }
            else {
                disablePanel("panelCategory");
                categoryId = select.value;
                return true;
            }
        }

        function callbackRating(button) {
            if (checkCategorySelected()) {
                rating = button.value;
                disablePanel("panelRatings");
                checkAndSubmitRating();
            }
        }

        function checkAndSubmitRating() {
            if (rating != undefined && experience != undefined
&& rating != "undefined" && experience != "undefined") {
                submitRating();
            }
            else {
                if (rating == undefined || rating ==
"undefined") {
                    showMessage("Please also give a rating to
the article.");
                }

                if (experience == undefined || experience ==
"undefined") {
                    showMessage("Please also answer 'Was this
information useful to you?' above question.");
                }
            }
        }

        function submitRating() {
            //alert("Submitting rating:" + rating + ";
experience:" + experience);
            setRating();
        }

        function setRating() {
            reqSetRating.open("GET", secureServerName + "/wrs-
webapp/wrs?method=setRating&username=" + username +
"&password=" + password + "&pageUrl=" + pageUrl +
"&categoryRating=" + categoryId + "&experience=" + experience +
"&rating=" + rating, true);
```

```
            reqSetRating.onload = processRequestSetRating;
            reqSetRating.send();
        }

        function processRequestSetRating() {
            var jsonValue = JSON.parse(reqSetRating.response);
            var success = true;
            if (jsonValue != null) {
                if (jsonValue.result != "success") {
                    showMessage(jsonValue.exception);
                    return;
                }
            }
            else {
                success = false;
            }


            if (success) {
                showMessage("Rating succesfully submitted");
            }
            else {
                showMessage("An error occurred while submitting
the rating");
            }
        }

        function showMessage(message) {
            var messageDiv =
document.getElementById("messageDiv");
            messageDiv.innerText = message;
        }

        function disablePanel(panel) {
            var trButtons = document.getElementById(panel);
            for (var i = 0; i < trButtons.children.length; i++)
{
                trButtons.children[i].disabled = true;
            }
        }

    </script>
</head>
<body onload="loaded();">
```

```
    <table>
        <tbody>
            <tr>
                <td align="center" colspan="2" style="font-
size: 20;">
                    Wikipedia Recommender System
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <table>
                        <tbody>
                            <tr>
                                <td align="right">
                                    <div id="loginText">
                                    </div>
                                </td>
                                <td style="font-weight: bold">
                                    <div id="divUsername">
                                    </div>
                                </td>
                                <td>
                                    <input id="loginButton"
type="button" value="Log out" onclick="NavigateToOptions();"
                                        style="width: 100px" />
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <table align="center">
                        <tbody>
                            <tr>
                                <td>
                                    Article rating:
                                </td>
                                <td style="font-weight: bold"
align="center">
                                    <div id="ratingDiv">
                                    </div>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <table align="center">
                        <tbody>
                            <tr>
                                <td>
                                    Category rating:
                                </td>
                                <td style="font-weight: bold"
align="center">
                                    <div
id="categoryRatingDiv">
                                    </div>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
            <tr>
                <td height="10">
                </td>
            </tr>
            <tr>
                <td style="font-size: 18; font-weight: bold"
colspan="2">
                    Your rating:
                </td>
            </tr>
            <tr>
                <td align="left">
                    <div>
                        Category</div>
                </td>
                <td id="panelCategory" align="center"
colspan="">
                    <select id="categoryComboBox">
                    </select>
                </td>
            </tr>
            <tr>
                <td align="left" colspan="2">
                    Was this information useful to you?
                </td>
            </tr>
            <tr>
                <td align="center" colspan="2"
id="panelExperience">
                    <button id="buttonNo"
onclick="callbackExperience(this);" value="false">
                        No
                    </button>
                    <button id="buttonYes"
onclick="callbackExperience(this);" value="true">
                        Yes
                    </button>
                </td>
            </tr>
            <tr>
                <td align="left" colspan="2">
                    Rate this article:
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <table border="0" cellpadding="0"
cellspacing="0" width="100%">
                        <tbody>
                            <tr>
                                <td>
                                    <table width="100%">
                                        <tr>
                                            <td
id="panelRatings">
                                                <button
id="button1" onclick="callbackRating(this);" value="1">
                                                    1
                                                </button>
                                                <button
id="button2" onclick="callbackRating(this);" value="2">
                                                    2
                                                </button>
                                                <button
id="button3" onclick="callbackRating(this);" value="3">
                                                    3
                                                </button>
                                                <button
id="button4" onclick="callbackRating(this);" value="4">
                                                    4
                                                </button>
                                                <button
id="button5" onclick="callbackRating(this);" value="5">
                                                    5
                                                </button>
                                                <button
id="button6" onclick="callbackRating(this);" value="6">
                                                    6
                                                </button>
                                                <button
id="button7" onclick="callbackRating(this);" value="7">
                                                    7
                                                </button>
                                                <button
id="button8" onclick="callbackRating(this);" value="8">
                                                    8
                                                </button>
                                                <button
id="button9" onclick="callbackRating(this);" value="9">
                                                    9
                                                </button>
                                            </td>
                                        </tr>
                                    </table>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
            <tr style="height: 50px; font-style: italic">
                <td colspan="2" align="center">
                    <div id="messageDiv" />
                </td>
            </tr>
        </tbody>
    </table>
```

```
</body>
</html>
```

options.html

```
<html>
<head>
    <title>WRS Options</title>
</head>
<script type="text/javascript" src="../global.js"></script>
<script type="text/javascript" src="../sjcl.js"></script>
<script type="text/javascript">

    var loginReq = new XMLHttpRequest();
    var createAccountReq = new XMLHttpRequest();

    function hash(password) {
        var result = sjcl.hash.sha256.hash(password);
        var hexValue = sjcl.codec.hex.fromBits(result);
        return hexValue;
    }

    function RestoreCredentials() {
        var username = localStorage[wrsUsernameString];
        var password = localStorage[wrsPasswordString];
        if (username == undefined || password == undefined ||
username == "undefined" || password == "undefined") {
            divLogin.style.display = "table-row";
            divLogout.style.display = "none";

            var textLoginUsername =
document.getElementById("textLoginUsername");
            var textLoginPassword =
document.getElementById("textLoginPassword")
            textLoginUsername.value = "";
            textLoginPassword.value = "";
        }
        else {
            divLogin.style.display = "none";
            divLogout.style.display = "table-row";

            var textLoggedInUsername =
document.getElementById("textLoggedInUsername");
            textLoggedInUsername.innerText = username;
        }

        if (localStorage[displayNotificationsString] == "true")
{
            checkboxDisplayNotifications.checked =
localStorage[displayNotificationsString];
        }
    }

    // Saves options to localStorage.
    function LogIn() {
        var textLoginUsername =
document.getElementById("textLoginUsername");
        var textLoginPassword =
document.getElementById("textLoginPassword");

        enableLoginPanel(false);

        login(textLoginUsername.value,
textLoginPassword.value);
    }

    function login(username, password) {
        loginReq.open("GET", secureServerName + "/wrs-
webapp/wrs?method=login&username=" + username + "&password=" +
password, true);
        loginReq.onload = processLoginRequest;
        loginReq.send();
    }

    function processLoginRequest() {
        var textLoginUsername =
document.getElementById("textLoginUsername");
        var textLoginPassword =
document.getElementById("textLoginPassword");

        var jsonValue = JSON.parse(loginReq.response);
        if (jsonValue != null) {
            if (containsException(jsonValue)) {
                alert(jsonValue.exception);
            }
            else {
                localStorage[wrsUsernameString] =
textLoginUsername.value;
                localStorage[wrsPasswordString] =
textLoginPassword.value;
                RestoreCredentials();

            }
        }
        else {
            alert("An error occured, please try again");
        }

        enableLoginPanel(true);
    }


    function enableLoginPanel(value) {
        var divLoginPanel =
document.getElementById("divLogin");
        divLoginPanel.disabled = !value;
    }


    // Restores select box state to saved value from
localStorage.
    function LogOut() {
        localStorage[wrsUsernameString] = undefined;
        localStorage[wrsPasswordString] = undefined;
        RestoreCredentials();
    }

    function CloseWindow() {
        window.close();
    }

    function ShowSignUp(display) {
        if (display == true) {
            divSignup.style.display = "table-row";
        }
        else {
            divSignup.style.display = "none";
        }
    }
    function SignUp() {
        createAccount(signUpUsername.value,
signUpPassword.value);
    }

    function changeNotificationRules() {
        localStorage[displayNotificationsString] =
checkboxDisplayNotifications.checked;
    }

    function createAccount(username, password) {
        createAccountReq.open("GET", secureServerName + "/wrs-
webapp/wrs?method=createUser&username=" + username +
"&password=" + password, true);
        createAccountReq.onload = createAccountReqCallback;
        createAccountReq.send();
    }

    function createAccountReqCallback() {
        var jsonValue = JSON.parse(createAccountReq.response);
        if (jsonValue != null) {
            if (containsException(jsonValue)) {
                alert(jsonValue.exception);
            }
            else {
                alert("Account succesfully created");
                clearCreateAccountFields();
                ShowSignUp(false);
            }
        }
    }

    function clearCreateAccountFields() {
        signUpUsername.value = "";
        signUpPassword.value = "";
    }


</script>
<body onload="RestoreCredentials();">
    <table width="400" style="background-color: #EEEEEE">
        <tr>
            <td align="left">
```

```
                <table style="background-color: #CCCCCC"
width="100%">
                    <tr>
                        <td style="font-size: 18; font-weight:
bold; color: Black;">
                            WRS
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td>
                <table>
                    <tr>
                        <td>
                            <input
id="checkboxDisplayNotifications" type="checkbox"
onchange="changeNotificationRules();" />Don't
                            display rating notification</br>
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr id="divLogin">
            <td>
                <table width="100%">
                    <tbody>
                        <tr>
                            <td colspan="2" align="right"
style="font-size: 18; font-weight: bold">
                                Login
                            </td>
                        </tr>
                        <tr>
                            <td align="right">
                                Username:
                            </td>
                            <td>
                                <input id="textLoginUsername"
type="text" style="width: 100%" />
                            </td>
                        </tr>
                        <tr>
                            <td align="right">
                                Password:
                            </td>
                            <td>
                                <input id="textLoginPassword"
type="password" style="width: 100%" />
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2">
                                <table width="100%">
                                    <tr>
                                        <td>
                                            <input
type="button" onclick="ShowSignUp(true);" value="Don't have an
account?" />
                                        </td>
                                        <td align="right">
                                            <input
type="button" onclick="LogIn();" value="Log in" style="width:
100px" />
                                        </td>
                                    </tr>
                                </table>
                            </td>
                        </tr>
                        <tr>
                            <!--  <td colspan="2">
                                <table width="100%">
                                    <tr>
                                        <td style="width: 100%"
align="right">

                                        </td>
                                        <td align="right">

                                        </td>
                                    </tr>
                                </table>
                            </td>-->
                        </tr>
                    </tbody>
                </table>
```

```
            </td>
        </tr>
        <tr id="divLogout">
            <td>
                <table width="100%">
                    <tbody>
                        <tr>
                            <td colspan="2" align="right"
style="width: 100%; font-size: 18; font-weight: bold">
                                Logged in
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <table width="100%">
                                    <tr>
                                        <td style="width:80px">
                                            Logged in as:
                                        </td>
                                        <td align="left">
                                            <div
id="textLoggedInUsername" style="font-weight: bold; width:
100%" />
                                        </td>
                                    </tr>
                                </table>
                            </td>
                            <td align="right">
                                <input id="buttonLogout"
type="button" onclick="LogOut();" value="Log out" style="width:
100px" />
                            </td>
                        </tr>
                    </tbody>
                </table>
            </td>
        </tr>
        <tr id="divSignup" style="display: none">
            <td>
                <table width="100%">
                    <tbody>
                        <tr>
                            <td colspan="2" align="right"
style="font-size: 18; font-weight: bold">
```

```
                                Sign up
                            </td>
                        </tr>
                        <tr>
                            <td align="right">
                                Username:
                            </td>
                            <td>
                                <input id="signUpUsername"
type="text" style="width: 100%" />
                            </td>
                        </tr>
                        <tr>
                            <td align="right">
                                Password:
                            </td>
                            <td>
                                <input id="signUpPassword"
type="password" style="width: 100%" />
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2" align="right">
                                <input type="button"
onclick="SignUp();" value="Sign up" style="width: 100px" />
                            </td>
                        </tr>
                    </tbody>
                </table>
            </td>
        </tr>
        <tr>
            <td align="center">
                <table style="height: 50px">
                    <tr>
                        <td>
                            <input type="button" value="Close
window" onclick="CloseWindow();" style="width: 100px" />
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
    </table>
```

```
</body>
</html>
```

backgroundPage.html

```
<html>
      <head>
        <script type="text/javascript"
src="../global.js"></script>
            <script language="javascript"
type="text/javascript">

                var wikipediaAddress =
'http://en.wikipedia.org/wiki/';
                loadCategories();

                chrome.tabs.onUpdated.addListener(function (id,
change, tab) {
                    if (change.status == "loading") {
                        if (IsValidWikipediaPage(tab)) {
                            var tabUrl = cleanAddress(tab.url);

                            initializeWikipediaPage(id, tabUrl,
getTitleFromUrl(tabUrl));
                        }
                        else {
                            initializeDifferentPage(id);
                        }
                    }

                    if (change.status == "complete") {
                    }
                });

            function IsValidWikipediaPage(tab) {
                var validPage = true;
                    var
beginning=tab.url.substr(0,wikipediaAddress.length);

                // exclude pages not belonging to wikipedia
                if(wikipediaAddress!=beginning)
                {
                    validPage = false;
                }
                // exclude special pages of wikipedia
                if (tab.url.indexOf("Special%3A") >= 0 ||
                    tab.url.indexOf("User:") >= 0 ||
                    tab.url.indexOf("Talk:") >= 0 ||
                    tab.url.indexOf("Wikipedia:") >= 0 ||
                    tab.url.indexOf("Help:") >= 0 ||
                    tab.url.indexOf("Media:") >= 0 ||
                    tab.url.indexOf("File:") >= 0 ||
                    tab.url.indexOf("Category:") >= 0 ||
                    tab.url.indexOf("action=history") >= 0 ||
                    tab.url.indexOf("action=edit") >= 0 ||
                    tab.url.indexOf("Portal:") >= 0 ||
                    tab.url.indexOf("Main_Page")>=0
                    ) {
                    validPage = false;
                }

                return validPage;
        }

            // the page is a wikipedia article
            function initializeWikipediaPage(id,pageUrl,title)
            {
                  chrome.pageAction.setIcon({path: "wrs
icon.png",tabId: id});
                  chrome.pageAction.setPopup({popup:
"popupPage.html",tabId: id})
                  chrome.pageAction.show(id);

            var username = localStorage[wrsUsernameString];
                  var password =
localStorage[wrsPasswordString];


      loadRatingForBackground(username,password,pageUrl,title);

}

            function initializeDifferentPage(id)
            {
                  //ignore other pages
                  chrome.pageAction.setIcon({path:
"icon.png",tabId: id});
```

```
                  chrome.pageAction.show(id);
            }

            </script>
      </head>
      <body>
      </body>
</html>
```

Bibliography

1. Adler, B. T. & Alfaro, L. d., 2007. *A Content-Driven Reputation System for the Wikipedia*, ACM Press.

2. Adler, T. B. & Alfaro, L. d., 2010. *Detecting Wikipedia Vandalism using WikiTrust,* Lab report for PAN at CLEF 2010.

3. Adler, T. B., Alfaro, L. d., Pye, I. & Raman, V., 2008. *Measuring Author Contributions to Wikipedia*, ACM Press.

4. Adler, T. B. et al., 2008. *Assigning Trust to Wikipedia Content*, ACM Press.

5. *Wikipedia.* [Online]
Available at: http://www.wikipedia.org/

6. bing.com, 2011. *Facebook Friends Now Fueling Faster Decisions on Bing.* [Online]
Available at:
http://www.bing.com/community/site_blogs/b/search/archive/2011/05/16/news-announcement-may-17.aspx?wa=wsignin1.0
[Accessed 01 08 2011].

7. Giles, J., 2005. *Special Report Internet encyclopaedias go head to head.* [Online]
Available at:
http://www.nature.com/nature/journal/v438/n7070/full/438900a.html
[Accessed 16 8 2011].

8. Jøsang, A., Ismail, R. & Boyd, C., 2007. *A Survey of Trust and Reputation Systems for online service provision.*

9. Jøsang, A., Keser, C. & Dimitrakos, T., 2005. *Can we manage trust?*, pp. 93-107.

10. Korsgaard, T. R., 2007. *Improving Trust in the Wikipedia,* Kongens Lyngby.

11. Lefevre, T., 2009. *Extending the Wikipedia Recommender System,* Kongens Lyngby.

12. Marsh, S. P., 1994. *Formalising Trust as a Computational Concept.*

13. Pilkauskas, P., 2010. *Expertise classification of recommenders in the Wikipedia Recommender System,* Kongens Lyngby.

14. Sanger, L., 2004. *Why Wikipedia Must Jettison Its Anti-Elitism.* [Online] Available at: http://www.kuro5hin.org/story/2004/12/30/142458/25 [Accessed 22 8 2011].

15. Schein, A., Popescul, A., Ungar, L. & Pennock, D., 2002. *Methods and Metrics for Cold-Start Recommendations.*

16. Victor, P., Cornelis, C., Teredesai, A. M. & De Cock, M., 2008. *Whom Should I Trust? The Impact of Key Figures on Cold Start Recommendations,* New York, pp. 2014-2018.