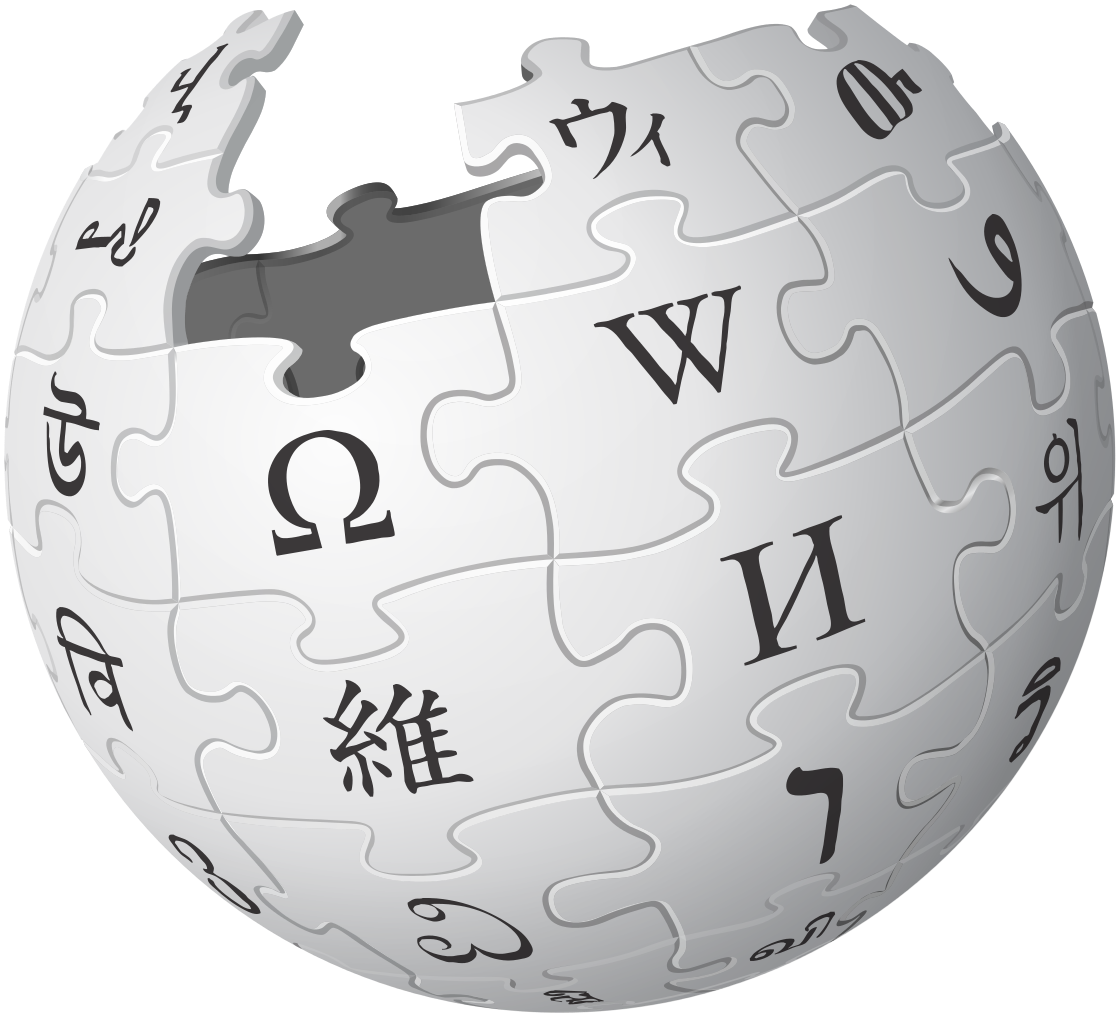
Title: WikiClassify™
Group Number: 11
Group Members: Nathan Kjer, Luke Wielgus, Adam Massoud, Brian Faure, Wanshu (Wayne) Sun, Brian Chu, Viswanathan Subramanian



Report 2 (Part 1)

Project Website: https://github.com/nathankjer/wikiclassify

Responsibility Matrix (Report 1)

	Adam	Nathan	Wayne	Brian F.	Luke	Brian C.	Viswanathan
Project Management (10 pts)	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Sec. 1: Customer Statement of Requirements (9 pts)	20%	20%	20%	20%	20%		
Sec. 2: System Requirements (6 pts)	20%	20%	20%	20%	20%		
Sec. 3: Functional Requirements (30 pts)	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Sec. 4: User Interface Specs (15 pts)	20%	20%	20%	20%	20%		
Sec. 5: Domain Analysis (25 pts)	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Sec. 6: Plan of Work (5 pts)	20%	20%			20%	20%	20%
Total Points:	16.3	16.3	15.3	15.3	16.3	10.3	10.3

Responsibility Matrix (Report 2)

	Adam	Nathan	Wayne	Brian F.	Luke	Brian C.	Viswanathan
Project Management (16 pts)							
Sec. 7: Interaction Diagrams (30 pts)	16.7%	16.7%	16.7%	16.7%		16.7%	16.7%
Sec. 8: Class Diagram & Interface Specification (10 pts)	16.7%	16.7%	16.7%	16.7%		16.7%	16.7%
Sec. 9: System Arch & System Design (15 pts)	16.7%	16.7%	16.7%	16.7%		16.7%	16.7%
Sec. 10: Algorithms & Data Structures(4 pts)							
Sec. 11: User Interface Design and Implementation (11 pts)							
Sec. 12: Design of Tests (12 pts)							
Sec. 13: Plan of Work (2 pts)							
Total Points:							

Table of Contents:

1. Customer Statement of Requirements (CSR) 5
   - 1.1. Problem Statement 5
   - 1.2. Solution 7
   - 1.3. Summary 8
   - 1.4. Glossary of Terms 8
2. System Requirements 12
   - 2.1. Enumerated Functional Requirements 12
   - 2.2. Enumerated Nonfunctional Requirements 13
   - 2.3. On-Screen Appearance Requirements 14
3. Functional Requirements Specifications 16
   - 3.1. Stakeholders 16
   - 3.2. Actors and Goals 16
   - 3.3. Use Cases 17
     - 3.3.1. Casual Description 17
     - 3.3.2. Use Case Diagram 18
     - 3.3.3. Traceability Matrix 19
     - 3.3.4. Fully-Dressed Description 19
   - 3.4. System Sequence Diagrams 21
4. User Interface Specification 24
   - 4.1. Preliminary Design 24
   - 4.2. User Effort Estimation 30
5. Domain Analysis 32
   - 5.1. Domain Model 32
     - 5.1.1. Concept Definitions 33
     - 5.1.2. Association Definitions 34
     - 5.1.3. Attribute Definitions 35
     - 5.1.4. Traceability Matrix 36
   - 5.2. System Operation Contracts 36
   - 5.3. Mathematical Model 37
6. Plan of Work 38
7. Interaction Diagrams 39
8. Class Diagram and Interface Specification 43
   - 8.1. Class Diagram 43
   - 8.2. Data Types and Operation Signatures 45
   - 8.3. Traceability Matrix 47
9. System Architecture and System Design 48
   - 9.1. Architectural Styles 48
   - 9.2. Identifying Subsystems 49
   - 9.3. Mapping Subsystems to Hardware 50
   - 9.4. Persistent Data Storage 50
   - 9.5. Network Protocol 51
   - 9.6. Global Control Flow 52
   - 9.7. Hardware Requirements 52
10. Algorithms and Data Structures
    - 10.1. Algorithms
    - 10.2. Data Structures
11. User Interface Design and Implementation
12. Design of Tests
13. Project Management and Plan of Work
    - 13.1. Merging the Contributions from Individual Team Members
    - 13.2. Project Coordination and Progress Report
    - 13.3. Plan of Work
    - 13.4. Breakdown of Responsibilities

1. Customer Statement of Requirements (CSR):

1.1 Problem Statement

Wikipedia is a crowdsourced online encyclopedia founded in 2001. With the goal of centralizing all of human knowledge, it attempts to be a neutral, global, and uncensored source of free information. As of 2016, Wikipedia is ranked number seven on the world's most popular websites, holds a top priority on search engines, and has been estimated[2] to be worth hundreds of billions of dollars. Furthermore, articles on Wikipedia have been cited[3] in hundreds of court cases. However, the most notable problem with Wikipedia is the quality of information in articles. Reliable or not, Wikipedia is one of the most heavily used online encyclopedias as a result of its high-quality content across a broad range of subjects.

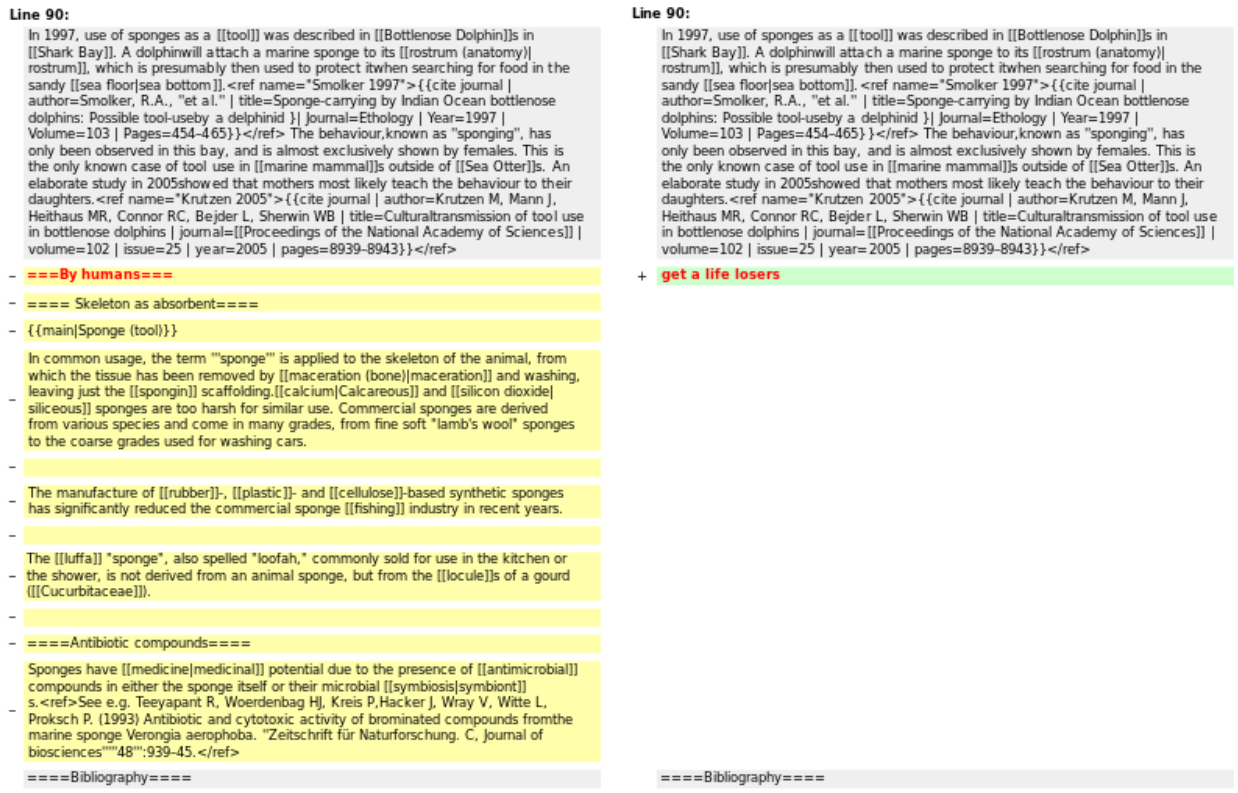
Anyone can edit Wikipedia, which is the primary reason why it is one of the largest online encyclopedias in the world. This results in almost an "infinite" source of information, and is widely used for schoolwork, research, and by anyone wishing to learn more about a certain subject or object. However, this has numerous tradeoffs. As anyone can edit articles, one can put misinformation or delete legitimate information on new or existing articles. In some cases, certain editors may not be as knowledgeable about the subject they are making edits about on the corresponding Wikipedia article. This will also result in misinformation, regardless of whether or not it was intentional. Destructive editing is a common problem that can go unnoticed for extended periods of time. There is even a list[4] that Wikipedia has created of bad article ideas, which is a result of destructive editing. Obviously, this decreases and sometimes destroys the condition and legitimacy of articles on Wikipedia. In addition, articles are frequently cited with unreliable sources, or not cited at all. This results in articles that may or may not be accurate and will negatively affect the credibility of articles and Wikipedia as a whole. Furthermore, Wikipedia has editor bias; 90% of Wikipedians are male, 40% of the world's population has access to the internet (Wikipedia can only be edited from a selection of 40% of the world's population), editors are often from white collar backgrounds, have stronger technical abilities, are more frequently from the Northern Hemisphere, and common languages and Western culture are more heavily represented. With such an editor bias, there is a larger probability of the content being misrepresented or not being of acceptable quality, as certain editors will input information that they feel is correct, but may only be a product of how opinionated they are about the subject they are editing about in the article.

Thus, although Wikipedia is one of the largely used online encyclopedias, despite having a substantial source of accurate articles, many articles are not of satisfactory quality due to the reasons stated above. Therefore, users of Wikipedia come across the problem of whether or not the article they are viewing has correct information. Being that Wikipedia is designed to provide users with information, any hint of inaccuracies or slight bias would be completely contradictory to the initial website intentions.

In order for users to combat this, software must be developed that can analyze and rate the quality of Wikipedia articles. This way, whenever a user views a Wikipedia

article, he or she will be able to tell if the information being viewed is of adequate quality or not. For the sake of convenience, having an extension or application created to implement this would be an ideal solution. From now on, we will refer to this software as WikiClassify. The idea is that whenever a user is viewing a wikipedia article, WikiClassify will activate. WikiClassify will analyze the content of the article and rate the quality of it, giving the user a good idea of whether or not he or she is looking at an article with quality information.

Wikipedia is a popular and useful online encyclopedia, but faces the issue of articles having information of poor quality. This is primarily due to destructive editing, cited sources that are not reputable, and the bias of the editors. In order to combat this, software will be created that will rate the quality of wikipedia articles, giving users a better idea of what they are looking at. The ultimate goal is for this software to be used not only for the benefit of Wikipedia users, but for the entire online encyclopedia to be "filtered" to the point that eventually all articles will be of satisfactory quality.



Line 90:

In 1997, use of sponges as a [[tool]] was described in [[Bottlenose Dolphin]]s in [[Shark Bay]]. A dolphinwill attach a marine sponge to its [[rostrum (anatomy)|rostrum]], which is presumably then used to protect itwhen searching for food in the sandy [[sea floor|sea bottom]].<ref name="Smolker 1997">{{cite journal | author=Smolker, R.A., "et al." | title=Sponge-carrying by Indian Ocean bottlenose dolphins: Possible tool-useby a delphinid }| Journal=Ethology | Year=1997 | Volume=103 | Pages=454–465}}</ref> The behaviour,known as "sponging", has only been observed in this bay, and is almost exclusively shown by females. This is the only known case of tool use in [[marine mammal]]s outside of [[Sea Otter]]s. An elaborate study in 2005showed that mothers most likely teach the behaviour to their daughters.<ref name="Krutzen 2005">{{cite journal | author=Krutzen M, Mann J, Heithaus MR, Connor RC, Bejder L, Sherwin WB | title=Culturaltransmission of tool use in bottlenose dolphins | journal=[[Proceedings of the National Academy of Sciences]] | volume=102 | issue=25 | year=2005 | pages=8939–8943}}</ref>

\- ===By humans===

\- ==== Skeleton as absorbent====

\- {{main|Sponge (tool)}}

\- In common usage, the term ""sponge"" is applied to the skeleton of the animal, from which the tissue has been removed by [[maceration (bone)|maceration]] and washing, leaving just the [[spongin]] scaffolding.[[calcium|Calcareous]] and [[silicon dioxide|siliceous]] sponges are too harsh for similar use. Commercial sponges are derived from various species and come in many grades, from fine soft "lamb's wool" sponges to the coarse grades used for washing cars.

\-

\- The manufacture of [[rubber]]-, [[plastic]]- and [[cellulose]]-based synthetic sponges has significantly reduced the commercial sponge [[fishing]] industry in recent years.

\-

\- The [[luffa]] "sponge", also spelled "loofah," commonly sold for use in the kitchen or the shower, is not derived from an animal sponge, but from the [[locule]]s of a gourd ([[Cucurbitaceae]]).

\-

\- ====Antibiotic compounds====

\- Sponges have [[medicine|medicinal]] potential due to the presence of [[antimicrobial]] compounds in either the sponge itself or their microbial [[symbiosis|symbiont]] s.<ref>See e.g. Teeyapant R, Woerdenbag HJ, Kreis P,Hacker J, Wray V, Witte L, Proksch P. (1993) Antibiotic and cytotoxic activity of brominated compounds fromthe marine sponge Verongia aerophoba. "Zeitschrift für Naturforschung. C, Journal of biosciences""48"":939-45.</ref>

====Bibliography====

Line 90:

In 1997, use of sponges as a [[tool]] was described in [[Bottlenose Dolphin]]s in [[Shark Bay]]. A dolphinwill attach a marine sponge to its [[rostrum (anatomy)|rostrum]], which is presumably then used to protect itwhen searching for food in the sandy [[sea floor|sea bottom]].<ref name="Smolker 1997">{{cite journal | author=Smolker, R.A., "et al." | title=Sponge-carrying by Indian Ocean bottlenose dolphins: Possible tool-useby a delphinid }| Journal=Ethology | Year=1997 | Volume=103 | Pages=454–465}}</ref> The behaviour,known as "sponging", has only been observed in this bay, and is almost exclusively shown by females. This is the only known case of tool use in [[marine mammal]]s outside of [[Sea Otter]]s. An elaborate study in 2005showed that mothers most likely teach the behaviour to their daughters.<ref name="Krutzen 2005">{{cite journal | author=Krutzen M, Mann J, Heithaus MR, Connor RC, Bejder L, Sherwin WB | title=Culturaltransmission of tool use in bottlenose dolphins | journal=[[Proceedings of the National Academy of Sciences]] | volume=102 | issue=25 | year=2005 | pages=8939–8943}}</ref>

\+ get a life losers

====Bibliography====

→ Comparison of a satisfactory Wikipedia article to an example of destructive editing
→ https://en.wikipedia.org/wiki/Reliability_of_Wikipedia[5]

1.2 Solution

At Wikipedia we utilize a rating system for our articles. This system allows Wikipedia users to automatically identify whether or not the article they are reading is verified by the Wikipedia community. The problem with this system is that it is extremely time-intensive to maintain and the process to apply for a better rating is long and tedious. Due to its inherent issues, we are seeking an automated approach to label and maintain labels for our entire library of articles.

Specifically we are looking for an algorithm which can use machine learning to implement the the assignment of featured, good, or null to every article following the criteria below:

https://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria[6]
https://en.wikipedia.org/wiki/Wikipedia:Good_article_criteria[7]

By analyzing the articles with the good and featured label, in tandem with the criterion above, we hope that you will be able to reverse engineer a 'formula' to label more articles accurately in the future.

Along with the rating system, Wikipedia also classifies articles into different subgroups. Classifications includes the following categories:

Maintenance category (Class)	Occurrence in text	Number of Articles	Notes
Stub Articles	stub}}	1954458	Article is very short and is not developed.
Cleanup	{{Cleanup	20678	Article requires general cleanup.
Advert	{{Advert	16462	Article is written like an advertisement.
Update	{{Update	13661	Article is outdated.
Tone	{{Tone	8307	Article does not match the tone of Wikipedia.
Featured Articles	{{Featured article}}	4659	Article is among the best on Wikipedia.
Plot	{{Plot	4090	Article is long/excessively detailed.
Essay	{{Essay-like	3716	Article is written like an opinion essay.
Peacock	{{Peacock	3555	Article overly promotes its subject.
Technical	{{Technical	3042	Article is overly technical.
Confusing	{{Confusing	2336	Article is confusing.
Overly Detailed	{{Overly Detailed	1724	Article is overly detailed.

→ Source: https://en.wikipedia.org/wiki/Wikipedia:Maintenance[8]

Having a system that is able to automatically classify every article and place them under a subgroup is very desirable. This type of system will help us target which articles need to be improved and the type of improvement that is needed. We believe the optimal solution would be either a chrome extension or a website, either of which would place the power of the machine learning process in the hands of the user. The specifics of the platform (precompiled database vs. real time algorithm etc.) are completely up to you, the only stipulation being that the interface should be quick and simple.

1.3 Summary

This project aims to rate and classify every article on Wikipedia using the known examples that have been assigned by our team. The features of the system should include the following:

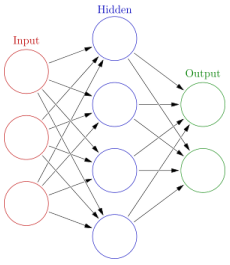
1. Input text and receive output parameters that classify the text based on our machine learning algorithm.
2. Uncover biases hidden in Wikipedia rating schema.
3. Compare accuracy of multiple input text articles.
4. Create procedurally generated articles on the fly.
5. Check text for historical accuracy/bias.
6. Compare different portions of Wikipedia based on article quality.

1.4. Glossary of Terms

Activation Function: The function that a neural network uses within its nodes.

Article: A single Wikipedia entry. Similar in structure and length to an article found in an encyclopedia.

Artificial Neural Network (ANN): Artificial neural networks, often simplified as “neural networks”, are a class of models used in machine learning. They are inspired by the mechanisms used by neurons in the brain.



An illustration of weight correlations within an ANN
Credit: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

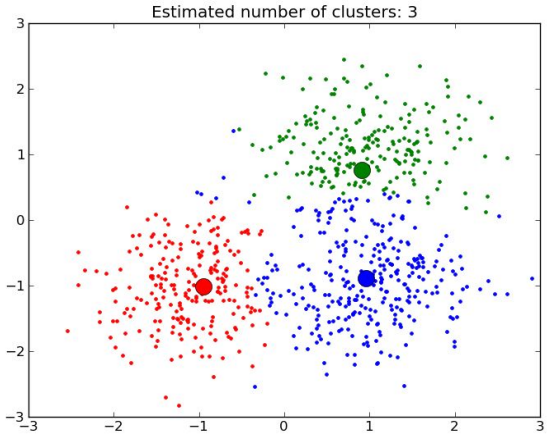
Backpropagation: A learning method that iteratively finds the gradient of the error to estimate a better solution of weights.

Bias: An underlying feature of text that skews it's accuracy according to the beliefs of the writer. An article with strong bias may present only partial facts and opinions in an attempt to sway the readers ideology towards that of the writer. An article without much bias will present information without undue weight to any side. This allows the reader to form their own opinions independent of the writer. Preventing article bias is extremely important to the Wikipedia community, as well as news sources and other credible sources of information.

Classification: The act of placing a selected sequence a category.

Class: A category. This is distinctly different than an OOP class.

Clustering: The relative grouping of article criteria pulled from the same classification, arises when the ML algorithm has found some relationship between the data that links articles of the same classifier. In a broad sense, clustering is the task of grouping objects in such a way that objects of the same cluster share more attributes than objects of differing clusters.



Clustering in two dimensions, with 3 classes
Credit: http://scikit-learn.sourceforge.net/0.5/modules/clustering.html

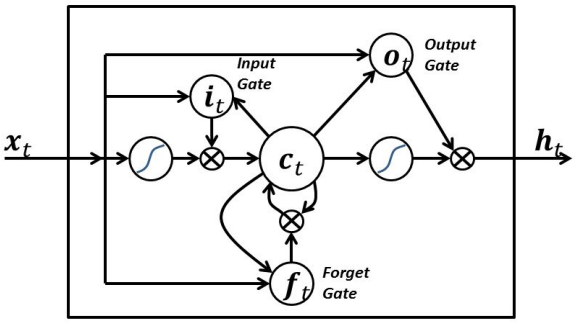
Error: The magnitude of difference between a model and its desired output.

Featured Article: A label given to certain Wikipedia articles. See the following link for more information: https://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria[6]

Google Chrome Extension: (or just "Chrome Extension") an extra piece of software that becomes part of the user's browser and allows it to run at the same time that the browser is running, allowing for new features to be added to the browser.

Login: The act of entering user credentials and being verified by the database.

Long short-term memory (LSTM): An artificial neural network architecture that is used to interpret and predict temporal data. It consists of a "memory cell" that contains a series of gates to try to solve the problem of vanishing weights.



Credit: https://commons.wikimedia.org/wiki/File:Long_Short_Term_Memory.png

Machine Learning: A programming technique wherein you provide the inputs and outputs and the algorithm does the grunt-work of finding a reliable relationship (or path) between the two. Effective if there is a large quantity of data pertaining to inputs and outputs. In our case, the inputs are the Wikipedia articles and the outputs are the article classifications. After an algorithm has been trained on a substantial quantity of input/output data, it can be used to predict outputs given only the inputs.

Noise: Data that only serves to hinder the progress of the algorithm. Noise pertains to, among other things, the html artifacts parsed out in the initial stages of the program.

Parse: The act of filtering and removing certain parts of something. In our case, we are sifting through the content of our data dumps and removing the html artifacts (formatting cues etc.)

Patterns: Relationships between certain pieces of data which serve to either substantiate, or invalidate prior assumptions. In our case, the ML algorithm finds underlying patterns in the articles which it can use as evidence that the article it is currently observing fits in some category.

Preprocessed Data: Classified text sequences (strings) are truncated into fixed lengths and converted into a numeric matrix format character-wise and preprocessed.

Raw Data: Pertains to the data directly after being downloaded and before passing through the parsing stage. At this point the data may contain junk formatting artifacts that must be removed.

Sampling: The process of extracting useful data from a body of text.

Supervised Learning: Machine learning method that builds a model based off of data with known inputs and outputs. This model can then predict outputs of new input data.

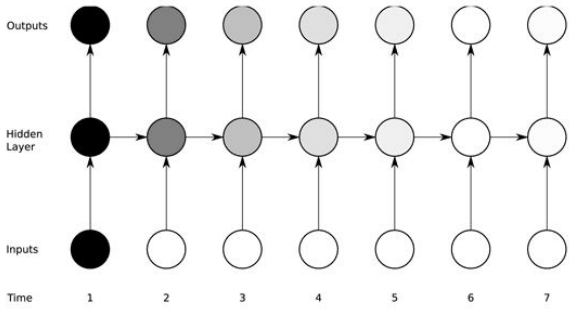
Tag: An element in the text which can be used to help in the identification process once it has been parsed.

Target Data: Consists of the Machine learning desired output data; algorithm is trained to make the connection from input data (text) to the output target data.

Text: Pertains to the ‘body’ of the Wikipedia article once it has been parsed.

Transformed Data: The text once all of the formatting artifacts have been removed; plain, lower-case text.

Vanishing Gradient Problem: An observed problem with machine learning algorithms, when models cannot retain information throughout their architecture or over time.



Credit: Graves, Alex. A Novel Connectionist System for Unconstrained Handwriting Recognition. 2009.

Visualization: How the results of the Machine learning algorithm can be viewed and interpreted by humans, this step is essential in making sense of the entire process.

2. System Requirements:

2.1 Enumerated Functional Requirements

Identifier	Requirement	PW
REQ1	The system shall use a model to classify sequences of text.	5
REQ2	The system shall use a model that can be trained, given examples of sequences with desired labels.	5
REQ3	The system shall have pre-labeled example data available, with methods for importing it.	4
REQ4	The system shall visualize sequence classifications for user interpretation.	4
REQ5	The system shall read in given text and classify based on given parameters.	4
REQ6	The system should allow the user to notify a Wikipedia administrator if an article is found to be of very poor quality.	2
REQ7	Given an input article (url or file), the system shall return various values associated with that article (either cached on server or compiled in real time).	3
REQ8	The system shall keep track of all registered users login information.	3

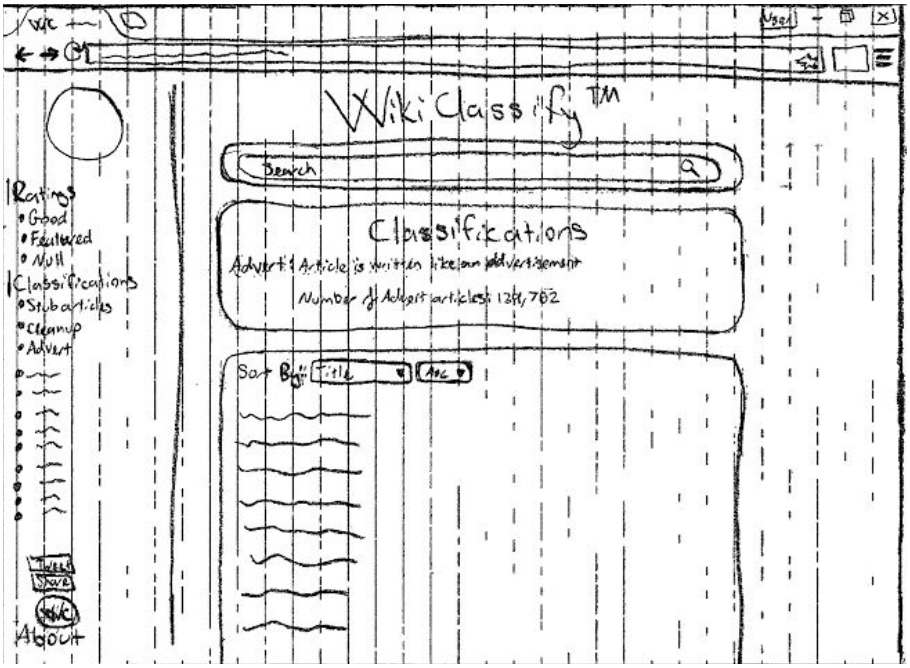
2.2 Enumerated Nonfunctional Requirements

Non-functional requirements are a more descriptive than practical listing the qualities of our system. These requirements are based on FURPS+, which are functionality, usability, reliability, performance, supportability, and other various attributes. These requirements are mainly concerned with quality attributes such as capability, compatibility, security, responsiveness, availability, efficiency, and maintainability.

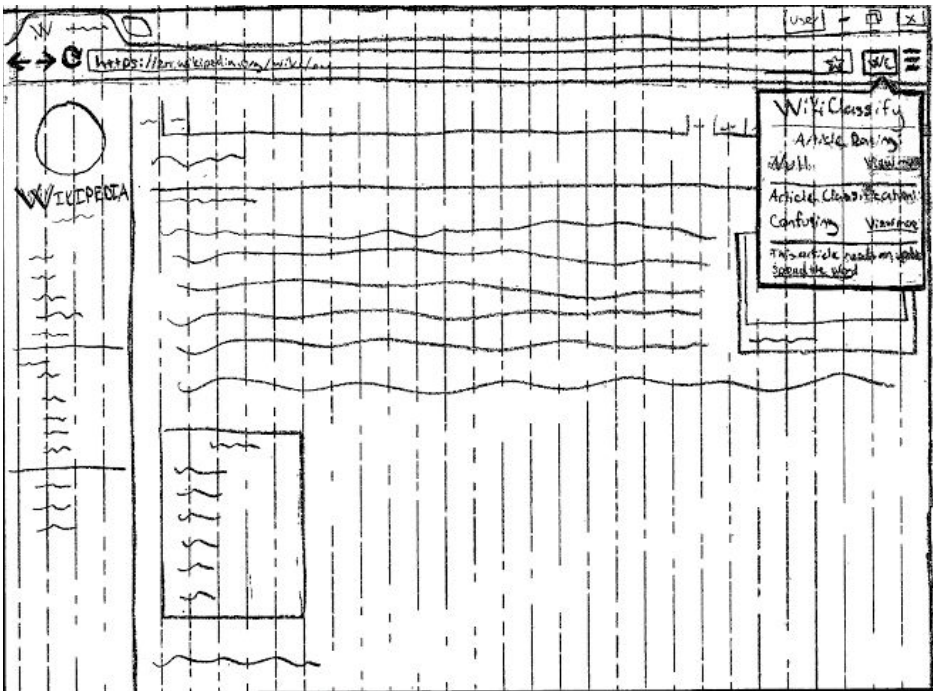
FURPS+ (User Stories)

Identifier	Requirement	PW
US 1	As a user, the system should be able to classify different articles found on Wikipedia.	5
US 2	As a user, I want the system to learn and be trained so that it can continue to work for articles written in the future.	5
US 3	As a user there should be documentation on how to use the program to the best of its abilities.	4
US 4	As a user I want a nice enough user interface so that I will be able to easily and quickly see data on the classification	4
US 5	As a user I would not want to be limited to just Wikipedia and instead would like to be able to apply the program to a wide range of texts.	4
US 6	As a user, if I find a problem with the article or the classification I should be able to notify an administrator to look into it.	2
US 7	As a user I would like the program to work quickly so that I do not have to wait for the program to run.	3
US 8	As an administrator, I would like to keep track of all registered users' information in order to avoid the use of bots on my website.	3

2.3 On-Screen Appearance Requirements



(Rough sketch of the user interface layout on the webpage)



(Rough sketch of the Chrome extension)

Identifier	Requirement
OSA1	Webpage: The webpage shall include an easy to maneuver layout. The left side of the page will include be the same for every page and will be static on scrolls. It will include the Wikipedia logo, the lists of the different ratings and classifications, social media sharing compatibility, our logo, and a link to our About page. The center of the page will always include “WikiClassify™” at the top along with the search. Depending on what current page you are on, the page will show the meaning of the rating or classification value you are searching through along with the number of articles for that rating or classification. Under that will the list of all the articles, where the user will also be able to sort the results.
OSA2	Chrome Extension (if time applicable): The Chrome extension should automatically pull the rating and classification of the current Wikipedia page from the server and display it as a popup. If the article needs to be updated, this extension will notify the user at the bottom and include a link for which the user can report the page.

3. Functional Requirements Specifications:

3.1 Stakeholders

1. Online encyclodpedias
2. Any user of the web services (registered or visitor)
3. Advertisers for the web services
4. Administrators for the web services

3.2 Actors and Goals

1. Online Encyclopedia
    - Type: Initiating
    - Goal: Pull data from database to better understand which articles need to be improved and why.
2. Registered User
    - Type: Initiating
    - Goal: Browse the WikiClassify website in order to see which how each article on Wikipedia is rated and classified. Registered users can also download the Chrome Extension to use while on Wikipedia.com and report missclassifications to the Administrator.
3. Visitor
    - Type: Initiating
    - Goal: Browse the WikiClassify website in order to see which how each article on Wikipedia is rated and classified.

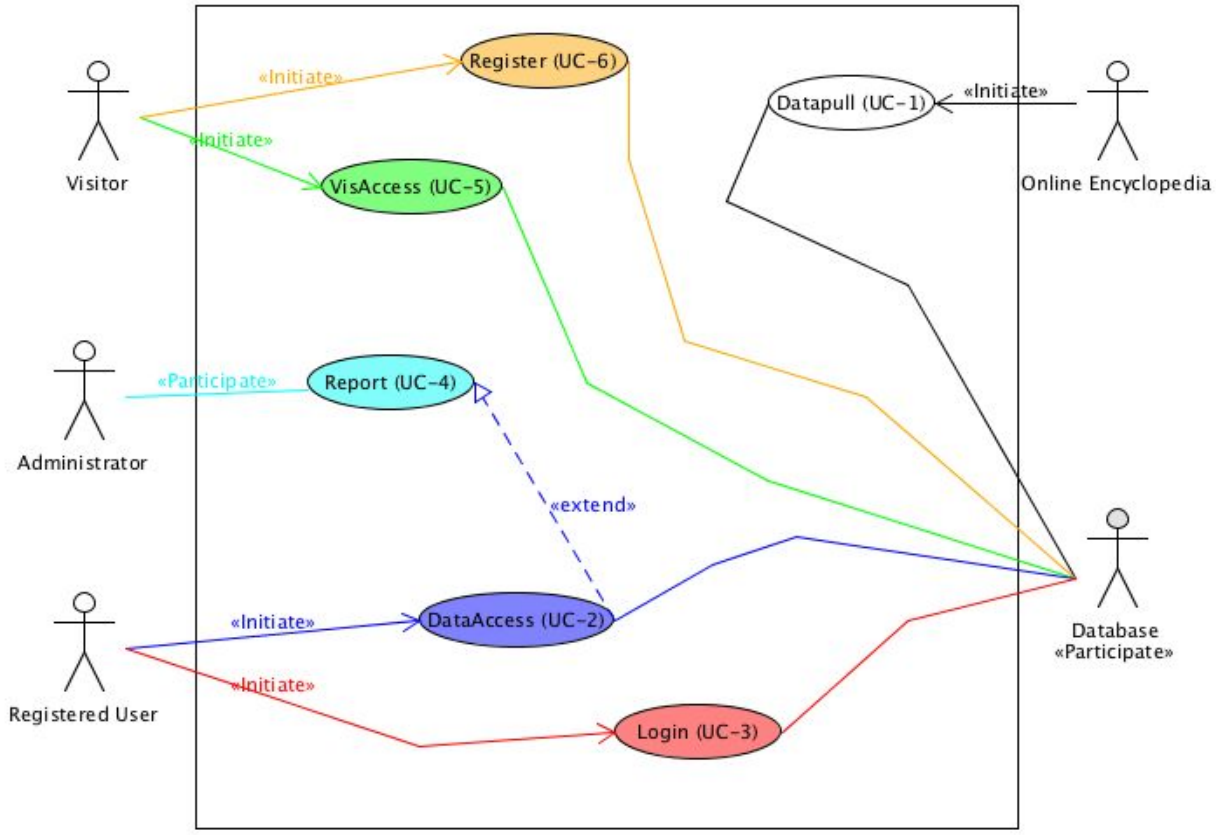
4. Administrator
    - Type: Initiating
    - Goal: Update and maintain database. Update and improve webservices.
5. Database
    - Type: Participating
    - Goal: Holds all the data and software to provide to the end users.

3.3 Use Cases

3.3.1 Causal Description

Actor	Actor's Goal	Use Case Name
Online Encyclopedia	Request data found from WikiClassify models to use for improvement of website	DataPull (UC-1)
Registered User	Access data to verify article accuracy while using Chrome extension or website	DataAccess (UC-2)
Registered User	Login to WikiClassify website	Login (UC-3)
Registered User	Report misclassified articles to Administrator	Report (UC-4)
Visitor	Access data on website	VisAccess (UC-5)
Visitor	Register for an account on WikiClassify	Register (UC-6)

3.3.2 Use Case Diagram



3.3.3 Traceability Matrix

REQ	PW	UC1	UC2	UC3	UC4	UC5	UC6
REQ1	5	X	X			X	
REQ2	5	X	X			X	
REQ3	4	X	X			X	
REQ4	4	X	X			X	
REQ5	4	X	X			X	
REQ6	2				X		
REQ7	3		X			X	
REQ 8	3			X			X
MAX PW		5	5	3	2	5	3
Total PW		22	25	3	2	25	3

3.3.4 Fully-Dressed Description

Use Case UC-1:
DataPull

Related Reqs: REQ1, REQ2, REQ3, REQ4, and REQ5
Initiating Actor: Online Encyclopedia
Actor's Goal: Request data found from WikiClassify models to use for improvement of website.
Participating Actors: Database
Preconditions: Data models aren't currently running/updating database.
Postconditions: Data is readily available to Online Ecyclopedia..
Flow of Events for Main Success Scenario:
-> 1.) **Online Encyclopedia** requests data pull through website to the **System**
<- 2.) **System** signals to the **Database** that the .zip file is being requested
-> 3.) **Database** directs **System** to .zip link
<- 4.) **System** directs **Online Encyclopedia** to link with the .zip file.
Flow of Events for Extensions (Alternate Scenarios):
3a. **Database** sends signal to **System** informing that it is running/updating, so it can not send data currently
<- 1.) **System** signals to **Online Encyclopedia** that the **Database** is currently running/updating and to try again later

Use Case UC-2: DataAccess

Related Reqs: REQ1, REQ2, REQ3, REQ4, REQ5, REQ6, and REQ7
Initiating Actor: Registered User
Actor's Goal: Access data to verify article accuracy while using Chrome extension or website.
Participating Actors: Database
Preconditions: Data models are currently available and user is logged in
Postconditions: Data integrity remains intact and data remains available to user

Flow of Events for Main Success Scenario:

-> 1. **Registered user** comes to website and requests to the **System** to access data on the site.
<- 2. **System** (a) acknowledges request, (b) **System** sends signal to **Database** to allow user to access data.
-> 3. **Database** approves user data access
<- 4. **System** grants user access to data
-> 5. **Registered user** searches for a specific article/title/url to access information.
<- 6. **System** (a) receives query, (b) **System** signals to **Database** to return webpage to user.
-> 7. **Database** (a) returns webpage to **System**, (b) which returns webpage to user.

Flow of Events for Extensions (Alternate Scenarios):

1a. **Registered User** uses the Chrome extension to search for a specific article/title/url to access information.
-> 1. **Registered user** requests to the **System** to access data on the Chrome extension.
<- 2. **System** (a) acknowledges request, (b) **System** sends signal to **Database** to allow user to access data.
-> 3. **Database** approves user data access
<- 4. **System** grants user access to data
-> 5. **Registered user** searches for a specific article/title/url to access information.
<- 6. **System** (a) receives query, (b) **System** signals to **Database** to return webpage to user.
-> 7. **Database** (a) returns webpage to **System**, (b) which returns webpage to user.

3a. **Database** sends signal to **System** informing that it is running/updating, so it can not send data currently
<- 1. **System** displays to **Registered user** an "error page link" message with a description of what is currently the issue for the request remaining unfilled and asks **Registered User** to try request again after a few minutes.

Use Case UC-5: VisAccess

Related Reqs: REQ1, REQ2, REQ3, REQ4, REQ5, REQ6, and REQ7
Initiating Actor: Visitor
Actor's Goal: Access data on website.
Participating Actors: Database
Preconditions: Data is currently viewable
Postconditions: Data integrity remains intact and data remains available to user

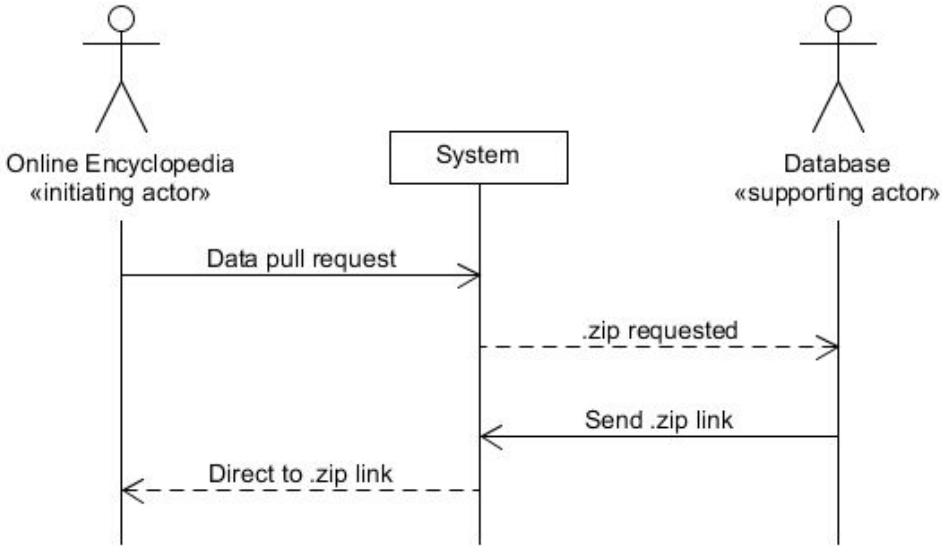
Flow of Events for Main Success Scenario:

-> 1.) **Visitor (a)** visits website to view data, (b) signals to the **System** to request information from the **Database**
<- 2.) **System** pulls data from the **Database**
-> 3.) **Database** sends data to the **System**
<- 4.) **System** displays data to the **Vistor** on the webpage

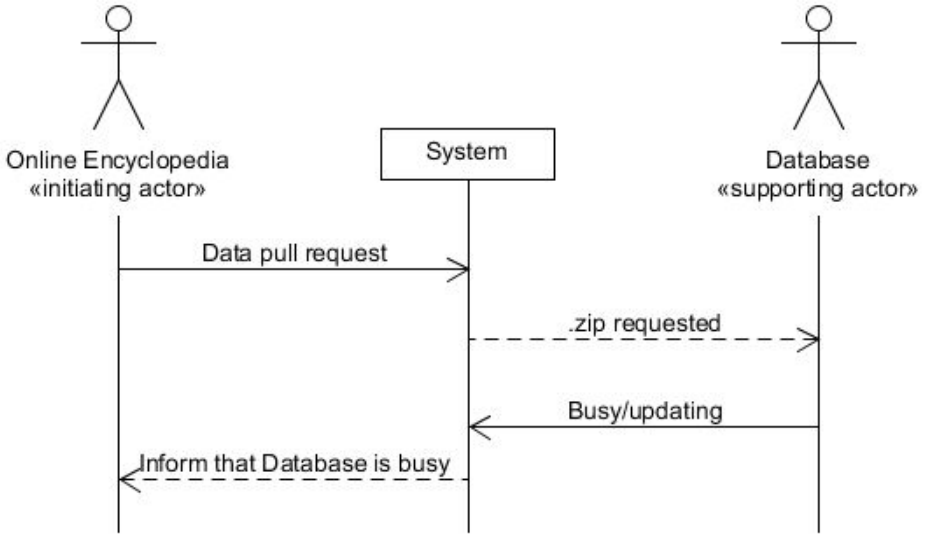
Flow of Events for Extensions (Alternate Scenario):

1a. **Database** sends signal to **System** informing that it is running/updating, so it can not send data currently
<- 1. **System** displays to **Visitor** an "error page link" message with a description of what is currently the issue for the request remaining unfilled and asks **Visitor** to try request again after a few minutes.

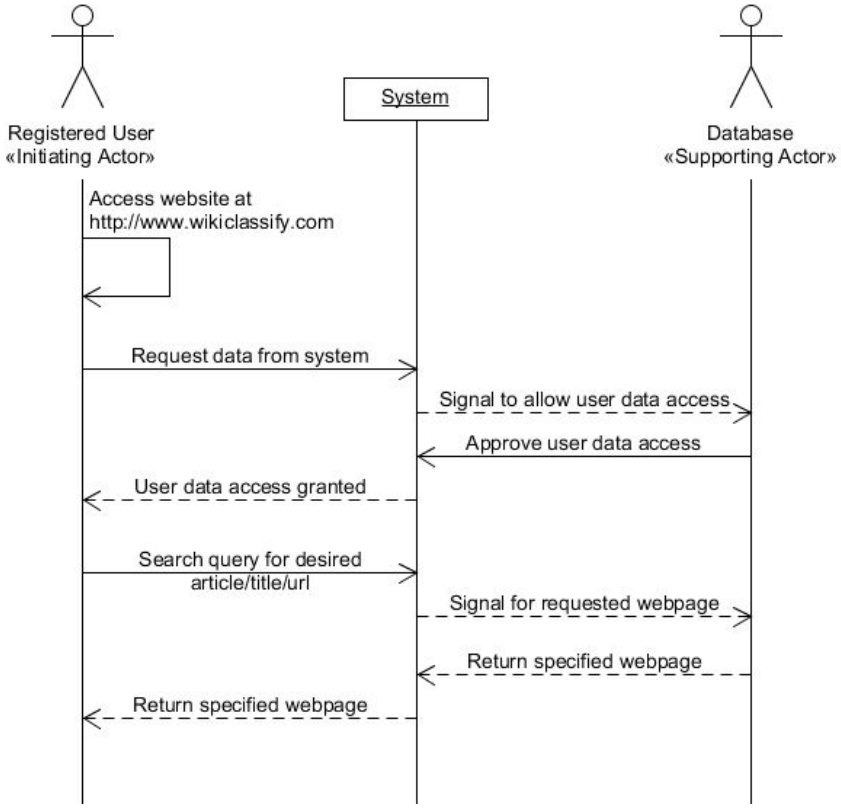
3.4 System Sequence Diagrams



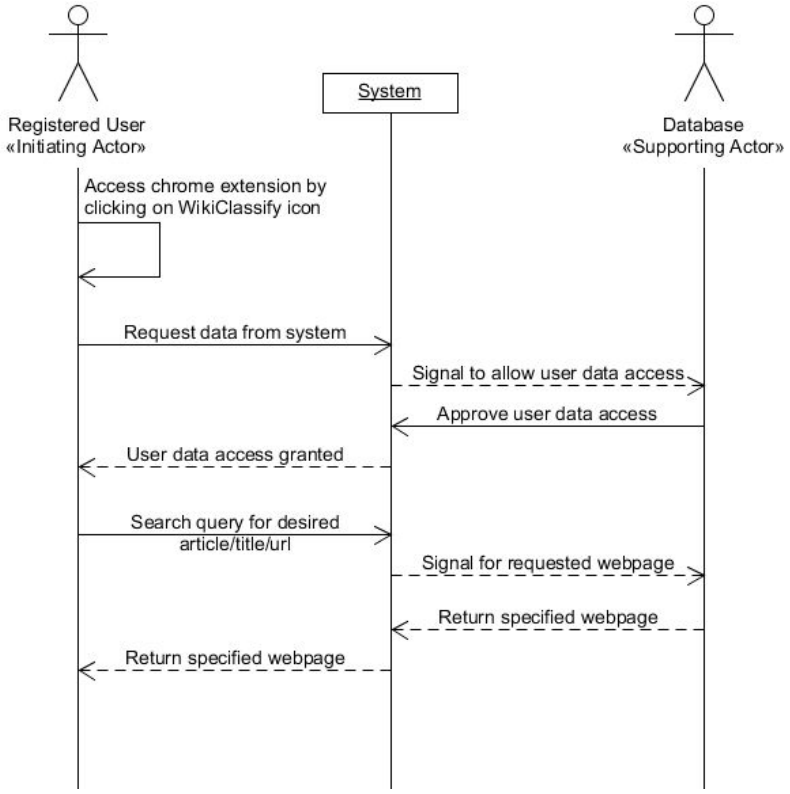
UC-1 (Main Success Scenario)



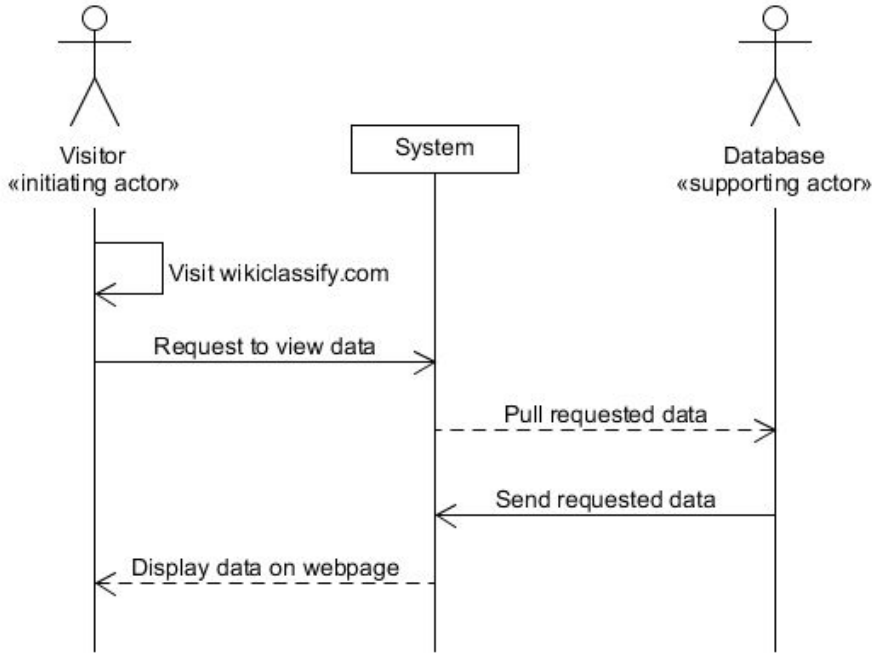
UC-1 (Alternate Scenario - Busy Database)



UC-2 (Main Success Scenario)



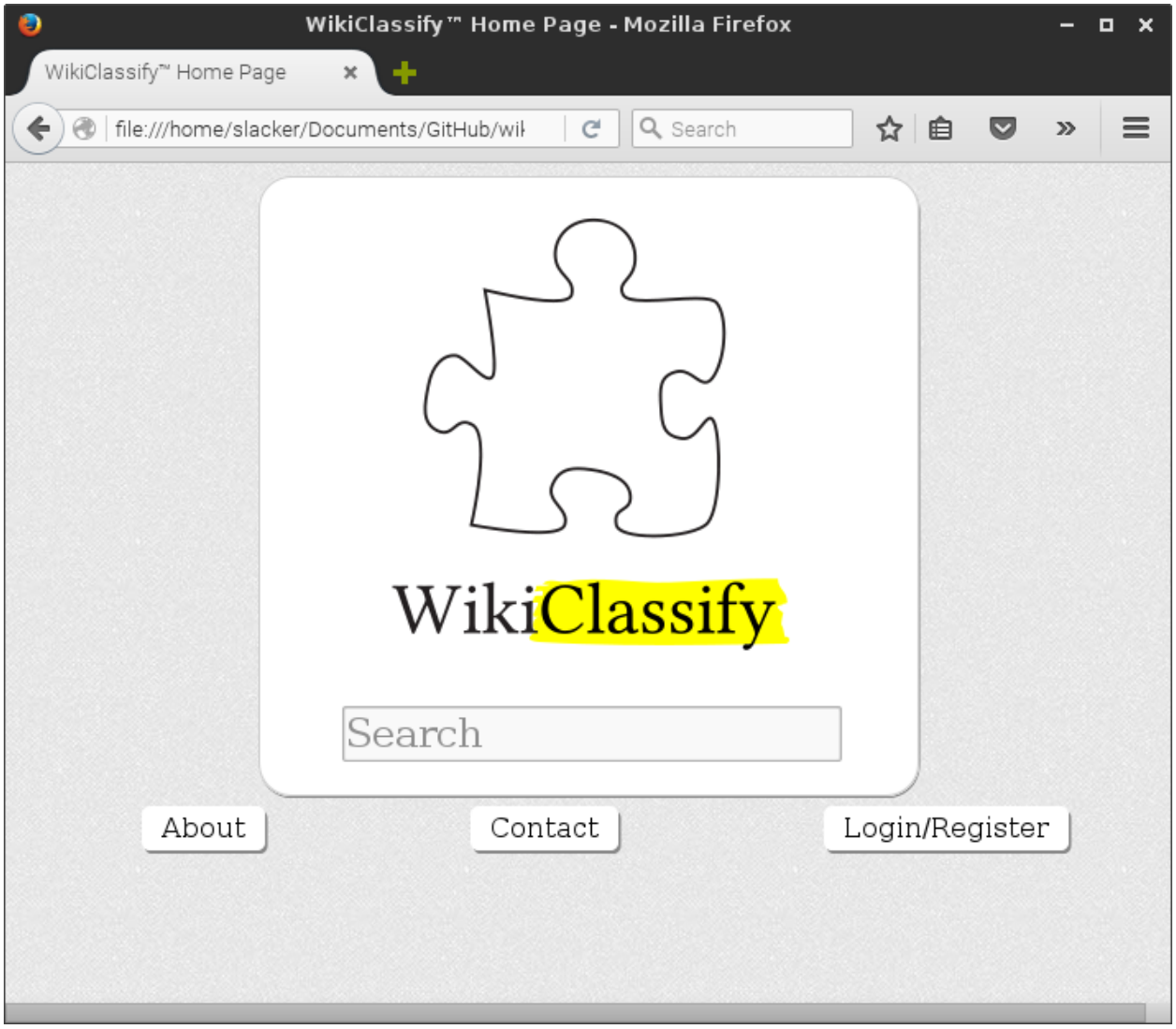
UC-2 (Alternate Scenario - Chrome Extension)



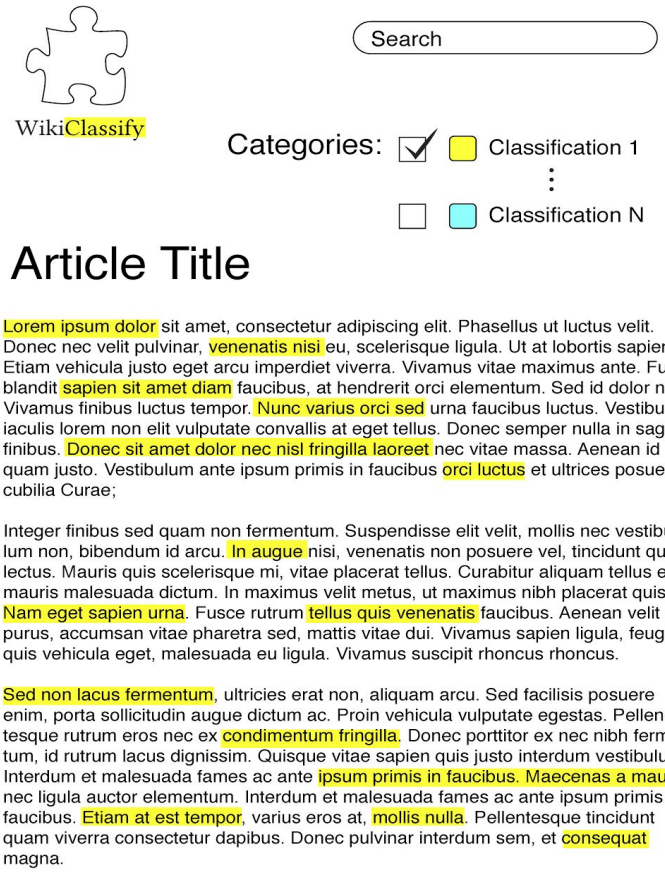
UC-5 (Main Success Scenario)

4. User Interface Specification:

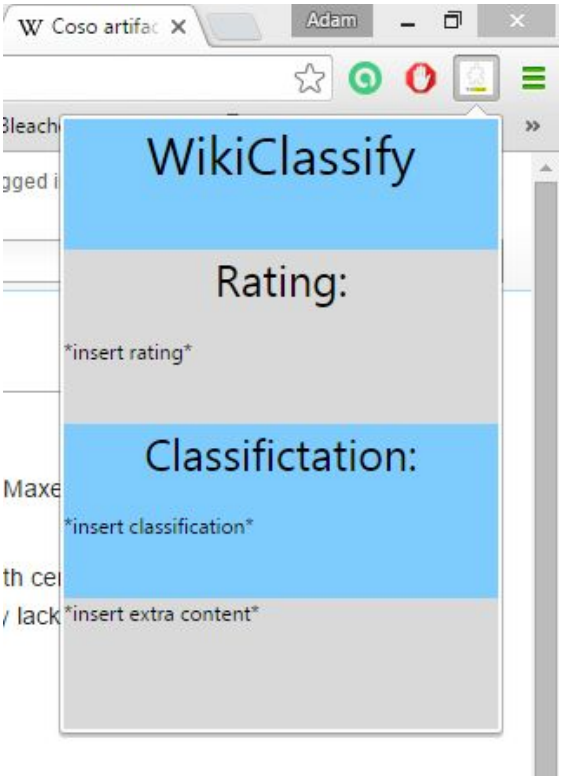
4.1 Preliminary Design



Basic homepage of the website set up to use our software. The webpage is very minimal and allows the user to learn about the software, contact us, or login to an account they registered at the links at the bottom of the page. To use the actual software the user would search for an article of their choice. Once the user types in the title or the URL of the article they wish to see they would press enter to continue to the next page.



Above is a mockup of a webpage after it has been chosen from search results. In the header is our logo and a search bar to find another article. Below the header are checkboxes to allow the user to choose what classification they would like to highlight, followed by the article with the character-wise highlighted text. The bulk of the pages on our site will be of this format.



Above is a mockup of a Google Chrome extension that furthers the functionality of our project. When on a webpage or Wikipedia article the user would be able to click on the extension and get quick stats about the page that they are currently on. With one click the user would see the rating of the article and the classification, along with any extra content about the page.

Summary of WikiPage file system and structure thus far…

→ **Base directory for wikipedia.cpp:**

#1.) *enwiki-20160113-pages-articles.xml*

→ Official wikipedia data dump (~50 GB)
→ XML format (similar to html)
→ Contains formatting artifacts along with the content we are parsing out
→ .xml file example (see extraneous formatting):

```
      <namespace key="2300" case="first-letter">Gadget</namespace>
      <namespace key="2301" case="first-letter">Gadget talk</namespace>
      <namespace key="2302" case="case-sensitive">Gadget definition</namespace>
      <namespace key="2303" case="case-sensitive">Gadget definition talk</namespace>
      <namespace key="2600" case="first-letter">Topic</namespace>
    </namespaces>
  </siteinfo>
  <page>
    <title>Africa</title>
    <ns>0</ns>
    <id>5334607</id>
    <revision>
      <id>703181583</id>
      <parentid>702065550</parentid>
      <timestamp>2016-02-04T00:16:49Z</timestamp>
      <contributor>
        <username>Seacactus 13</username>
        <id>17079922</id>
      </contributor>
      <minor/>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <text xml:space="preserve" bytes="107284">{{Other uses}}
{{pp-semi-indef}}
{{Infobox Continent
|title = Africa
|image = [[File:Africa (orthographic projection).svg|200px]]
|area = {{convert|30,221,532|km2|sqmi|abbr=on}},&amp;nbsp;[[List of continents by area|2nd]]
|population = 1.1 billion&lt;ref name=&quot;2013 World Population Data Sheet&quot;&gt;{{cite web |url=http://www
prb.org/pdf13/2013-WPDS-infographic_MED.pdf |title=2013 World Population Data Sheet |last=Gudmastad |first=Erich
|date=2013 |website=www.prb.org |publisher=[[Population Reference Bureau]] |access-date=18 August 2015 }}&lt;/re
gt; (2013, [[List of continents by population|2nd]])
|density = {{pop density|1100000000|30221532|km2|sqmi}}
|demonym = [[Ethnic groups of Africa|African]]
|countries = 54 (and 2 disputed)
|list_countries = List of sovereign states and dependent territories in Africa
|dependencies = {{Collapsible list
 | list_style = text-align:left;
 |title = External (3)
 | 1 = [[Mayotte]] | 2 = [[Réunion]] | 3 = [[Saint Helena, Ascension and Tristan da Cunha]]
 }}
 {{Collapsible list
 | list_style = text-align:left;
```

#2.) *wikipedia.cpp*

→ C++ file
→ Fetches text from the .xml file, parses it into individual wikipedia articles, parses out information from each articles, saves to parsed directory

→ **parsed directory:**

#1.) *hashfile.txt*

hashfile.txt

```
Cache Date: Fri Feb 12 17:41:04 2016

[AccessibleComputing] redirect/vol-0.txt
[Anarchism] good/vol-0.txt
[AfghanistanHistory] redirect/vol-1.txt
[AfghanistanGeography] redirect/vol-2.txt
[AfghanistanPeople] redirect/vol-3.txt
[AfghanistanCommunications] redirect/vol-4.txt
[AfghanistanTransportations] redirect/vol-5.txt
```

→ Contains every wikipedia article title along with the sub-directory it has been saved under
→ (~200 MB)

#2.) *bad, good, redirect & regular*

→ bad = Articles which contain some form of cleanup template (tagged by wikipedia users)
→ good = Articles which contain good or featured tag.
→ redirect = Articles that redirects to a different article, such as a misspelled title that points you to the correct article
→ regular = Articles that don't fit into any of the other three directories.

→ Example file:

vol-3.txt — Open with TextEdit

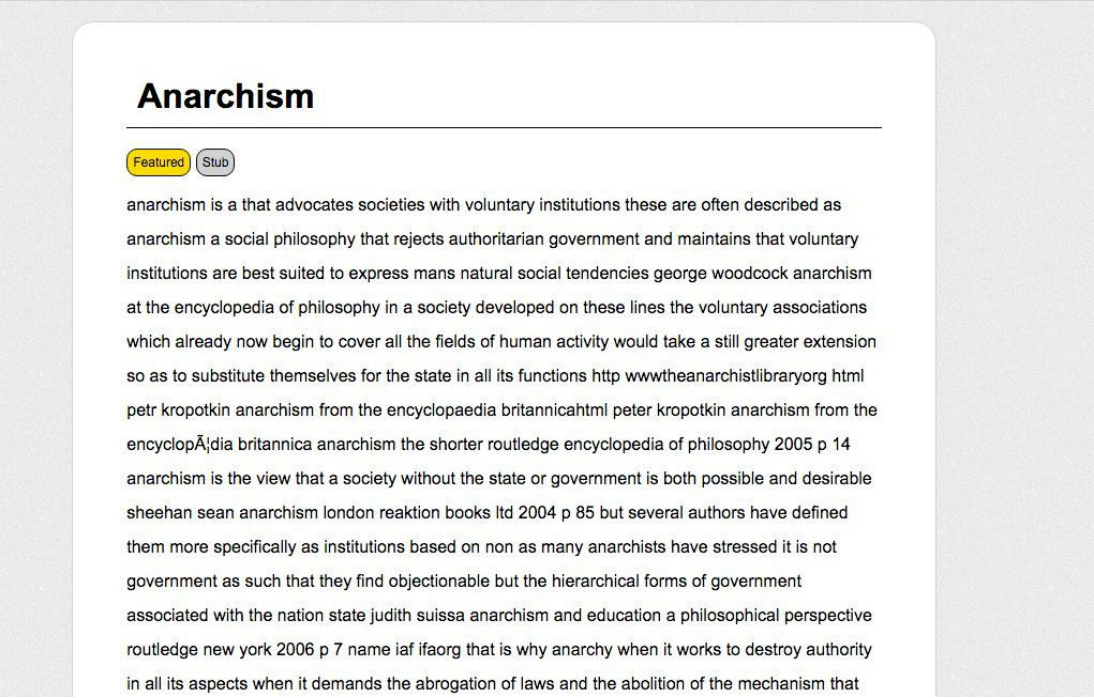
```
---> VERSION 1.0
Title:          Apollo 11
Namespace:      0
Article size:   37150
Redirect:       0
Redirection:
Quality:        2
Contributor:    CAPTAIN RAJU
Timestamp:      2016-01-10T19:21:37Z
Pic Count:      32
Template:

apollo 11 was the first  that  humans on the  americans  and  landed on july 20 1969 at 2018   years ago armstrong
became the first to step onto the lunar surface six hours later on july 21 at 0256 utc armstrong spent about two and a
half hours outside the spacecraft and together with aldrin collected  of lunar material for return to earth the third
member of the mission  piloted the  alone in lunar orbit until armstrong and aldrin returned to it just under a day
later for the trip back to earthlaunched by a  rocket from  in  on july 16 apollo 11 was the fifth manned mission of s
the apollo  had three parts a  cm with a cabin for the three astronauts and the only part that landed back on earth a
sm which supported the command module with propulsion electrical power oxygen and water and a  lm for landing on the
moon which itself was composed of two parts after being sent toward the moon by the saturn vs upper stage the
astronauts separated the spacecraft from it and traveled for three days until they entered into lunar orbit armstrong
and aldrin then moved into the lunar module and landed in the  they stayed a total of about 21½ hours on the lunar
surface after lifting off in the upper part of the lunar module and rejoining collins in the command module they
returned to earth and landed in the pacific ocean on july 24broadcast on live tv to a world wide audience armstrong
stepped onto the lunar surface and described the event as one small step for a man one giant leap for mankind apollo 11
effectively ended the  and fulfilled a national goal proposed in 1961 by the  in a speech before the  before this
decade is out of landing a man on the moon and returning him safely to the earth frameworkapollo 11 was the second all
veteran multi person crew the first being  in human spaceflight history  a previous solo veteran flight had been made
on soyuz 1 in 1967 by  cosmonaut ryan p 1969 the invasion of the moon 1969 the story of apollo 11 penguin books
middlesex england collins was originally slated to be the command module pilot cmp on  but was removed when he required
surgery on his back and was replaced by  his backup for that flight after collins was medically cleared he took what
would have been lovells spot on apollo 11 as a veteran of apollo 8 lovell was transferred to apollo 11s backup crew but
promoted to backup commanderin early 1969 anders accepted a job with the  effective august 1969 and announced that he
would retire as an astronaut on that date at that point  was moved from the support crew into parallel training with
anders as backup command module pilot in case apollo 11 was delayed past its intended july launch at which point anders
would be unavailable if needed and would later join lovells crew and ultimately be assigned as the original  cmp
capcom capcom capcom capcom capcom capcomcliff charlesworth green team launch and  white team lunar landing black team
lunar ascentafter the crew of apollo 10 named their spacecraft charlie brown and snoopy assistant manager for public
affairs julian scheer wrote to  director  to suggest the apollo 11 crew be less flippant in naming their craft during
```

→ Each folder (good, bad, redirect, regular) contains thousands of .txt files, similar to the one above. All formatting has been removed for the most part (compare to the .xml file example above).

→ This filesystem will allow us to pull the wikipedia articles into our Python script which will eventually run the machine learning algorithm.

→ In the future we may consider a binary or hex format to save on memory and cpu time and possibly a binary tree type format for accessing the files.



Above is an unclassified webpage made during a preliminary draft of our code. It is the result of data read from the dump and rewritten it into an html template. Below the article title are ground truth labels associated with the article.

4.2 User Effort Estimation

Requesting Data Found From WikiClassify Models to Use for Improvement of Website

1. Search for article title of length n in search bar on the WikiClassify website - (Keystrokes: n + 1 // Mouse Clicks: 0)
    a. Enter article title in search bar
    b. Press “Enter”

    ----Software will analyze the article and display data about it----

Access Data to Verify Article Accuracy (Chrome Extension or Website)

1. Search for article title of length n in search bar on the WikiClassify website - (Keystrokes: n + 1 // Mouse Clicks: 0)
    a. Enter article title in search bar
    b. Press “Enter”

    ----Software will analyze the article and display data about it----

Login to WikiClassify Website

1. Access login screen - (Keystrokes: 0 // Mouse Clicks: 1)
    a. Click “login” button
2. Login into account - (Keystrokes: u + p + 1 // Mouse Clicks: 1)
    a. Enter username of length u in “Username:” section
    b. Click on “Password:” section
    c. Enter password of length p in “Password:” section
    d. Press “Enter”

    ----User will be brought back to home screen, but will be logged into his/her account---

Report Misclassified Articles

----User is assumed to be viewing an article at the time of this use case----

1. Report the article - (Keystrokes: 0 // Mouse Clicks: 1)
    a. Click “Report Article” button

    ----User will be brought to page for reporting articles----
2. Specify what the article error is - (Keystrokes: 0 or 0 + c // Mouse Clicks: 2 or 4)
    a. Click on pulldown menu
    b. Select “Misclassified Article”
    c. ***Optional*** Click on “Comments” section
    d. ***Optional*** Enter in comments of length c about misclassified article (if any)

    ----User will have successfully reported the article for misclassification, and will be brought back to the article page----

Access Data on Website (as a visitor)

1. Search for article title of length n in search bar on the WikiClassify website - (Keystrokes: n + 1 // Mouse Clicks: 0)
    a. Enter article title in search bar
    b. Press “Enter”

    ----Software will analyze the article and display data about it----

Register for an Account on WikiClassify

1. Access account registration page (Keystrokes: 0 // Mouse clicks: 1)
    a. Click “Register” button

    ----User will be brought to the account registration page----
2. Create an account (Keystrokes: e + 2p + 1 // Mouse Clicks: 2)
    a. Enter in email address of size e
    b. Click on “Password:” section
    c. Enter in password of size p
    d. Click on “Confirm Password”
    e. Enter in password of size p
    f. Press “Enter”

    ----User will have successfully created an account, logged in, and will be brought back to the home screen of the website----

Send a Request to Place an Advertisement on WikiClassify Website

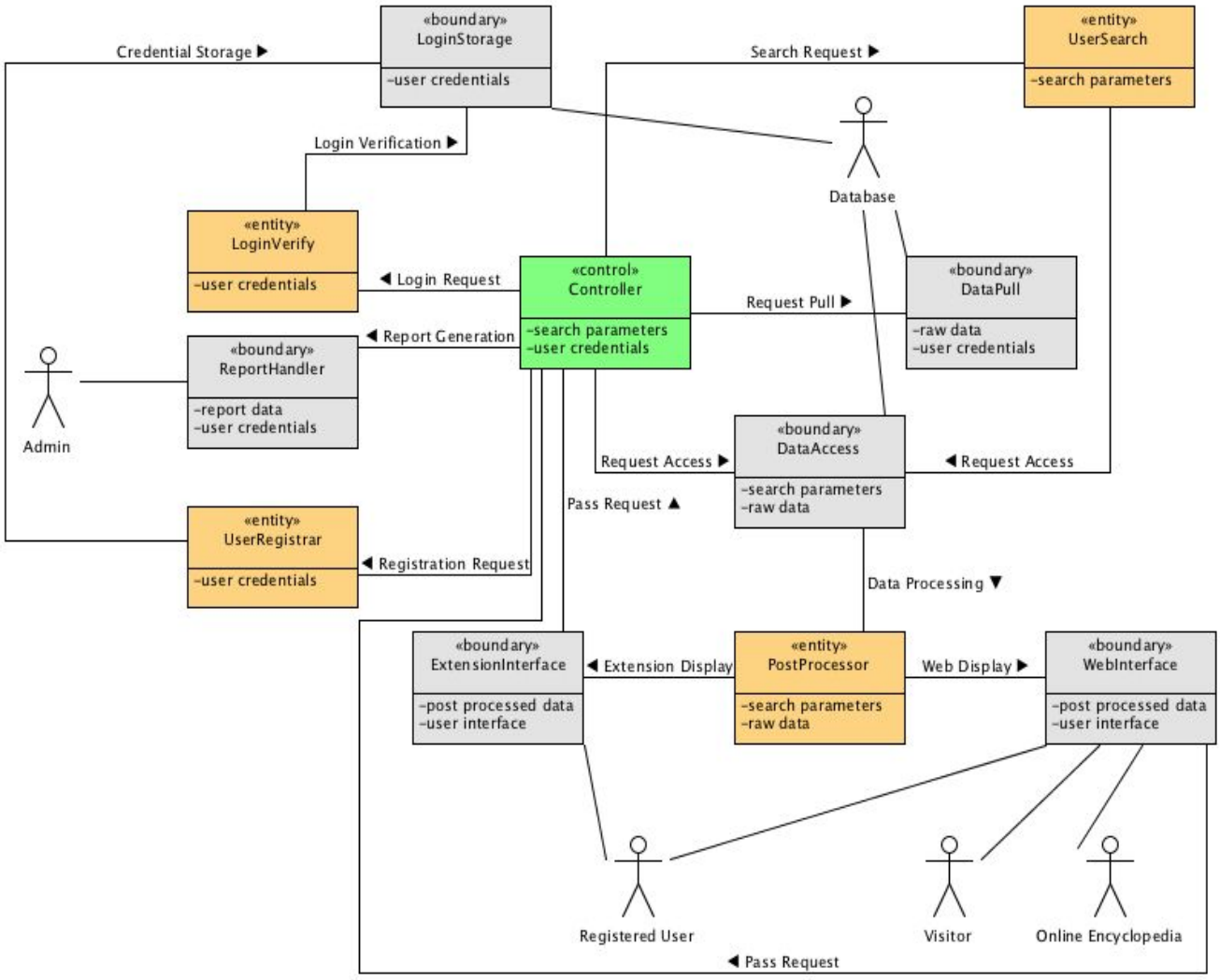
1. Access the “Contact Us” page (Keystrokes: 0 // Mouse Clicks: 1)
    a. Click on the “Contact Us” button

    ----User will be brought to page where users’ can contact the WikiClassify team----
2. Submit advertisement request (Keystrokes: n + e + c // Mouse Clicks: 4)
    a. Click on “Name:” section
    b. Enter in name of size n
    c. Click on “E-mail:” section
    d. Enter in email of size e
    e. Click on “Comments:” section
    f. Enter in comments of size c (this is where the advertiser will enter in his/her request for an advertisement)
    g. Click “submit” button

    ----User will have successfully submitted a request for an advertisement. The WikiClassify team will respond to user’s request via email----

5. Domain Analysis

5.1 Domain Model



5.1.1 Concept Definitions

Responsibility Description	Type	Concept Name
Coordinate actions of all concepts associated with a use case, a logical grouping of use cases, or the entire system and delegate the work to other concepts.	D	Controller
caller from pull data from repository	D	DataPull
caller views data from DataPull	D	DataAccess
Verification to login to website	D	LoginVerify
Intermediary between user reports, bugs, etc. and administrators	D	ReportHandler
A way to register a user account to allow the extra functionality of having an account	D	UserRegistrar
A browser extension that adds extra functionality to the basic browser. The extension would extend the functionality of the website to allow the the user to use the program without having to be on the website	D	ExtensionInterface
A webpage that allows the user to interact with the program	D	WebInterface
Used for filtering searches	D	PostProcessor
Allows the user to search for an article on the web interface	D	UserSearch
A small database to hold all login information	K	LoginStorage

5.1.2 Association Definitions

Concept Pair	Association Description	Association Name
Controller ↔ DataPull	Controller sends request for DataPull to pull data and send data to DataAcess	Request Pull
Controller ↔ DataAccess	Controller informs Data Acess to be ready to receive data	Request Access
Controller ↔ UserSearch	Controller processes and receives user request and sends out appropriate commands	Search Request
UserSearch ↔ DataAccess	UserSearch sends search criteria to data Access and receives post processed data	Search Data
DataAccess ↔ PostProcessor	DataAcess shares search credentials and raw data	Data Processing
PostProcessor↔ ExtensionInterface	ExtensionInterface sends request for data to display and receives it from postProcessor	Extension Display
PostProcessor↔ WebInterface	WebInterface sends request for data to display and receives it from postProcessor	Web Display
Controller↔ LoginVerify	Controller sends user login request to begin login process	Login Request
LoginVerify↔ LoginStorage	LoginVerify sends user credentials to LoginStorage to be verified,LoginStorage sends back results	Login Verification
Controller↔ ReportHandler	ReportHandler signals to controller that error is detected and report is generated	Report Generation
Controller↔Registrar	Controller sends request and user credentials to be registred	Registration Request
UserRegistrar↔ LoginStorage	Registrar sends user credentials and LoginStorage updates database	Credential Storage
WebInterface ↔ Controller	Pass user input request to the controller to allow for processing	Pass Request
ExtensionInterface ↔ Controller	Pass user input request to the controller to allow for processing	Pass Request

5.1.3 Attribute Definitions

Concept	Attribute	Attribute Description
Controller	search parameters	User provided criteria to locate data including type,length,name
	user credentials	Data provided to identify and verify user to grant proper access rights
DataPull	raw data	Unprocessed data pulled from repository
	user credentials	Sent from controller,checked for authenticity to verify user
DataAccess	search parameters	Copied from controller,used to find desired data
	raw data	Data received from DataPull to be processed
LoginVerify	user credentials	Sent from controller,checked for authenticity to verify user
ReportHandler	report data	Data containing report attributes such as time.
	user credentials	Copied from controller, Identifies the user making the report
UserRegistrar	user credentials	Copied from controller, Identifies user to be added to database
ExtensionInterface	post-processed data	Processed data to be displayed from post processor
	user interface	Processes user commands and sends it to controller
WebInterface	post-processed data	Processed data to be displayed from post processor
	user interface	Processes user commands and sends it to controller
PostProcessor	search parameters	Shared with DataAcess to enable data filter for desired data
	raw data	Unprocessed data to be filtered received from data access
UserSearch	search parameters	Article title or URL
LoginStorage	user credentials	Credentials to be stored

5.1.4 Traceability Matrix

UC	PW	Controller	DataPull	DataAccess	LoginVerify	ReportHandler	UserRegistrar	ExtensionInterface	WebInterface	PostProcessor	UserSearch	LoginStorage
UC 1	22	X	X		X							X
UC 2	25	X		X				X	X	X	X	
UC 3	3	X			X							X
UC 4	2	X				X						X
UC 5	25	X		X				X	X	X	X	
UC 6	3	X					X					X

5.2 System Operation Contracts

Operation	DataPull
Preconditions	• login credentials verified and they have permissions to pull the data • dataBaseActive == false
Postconditions	• data is available to be downloaded (.zip file)

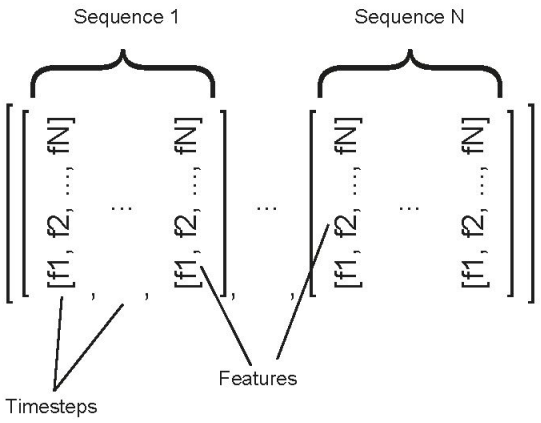
Operation	DataAccess
Preconditions	• login credentials verified • dataBaseActive == false • searchUsed == true, searchParameters • searchUsed == false
Postconditions	• data is displayed on screen

Operation	VisAccess
Preconditions	• dataBaseActive == false • searchUsed == true, searchParameters • searchUsed == false
Postconditions	• data is displayed on screen

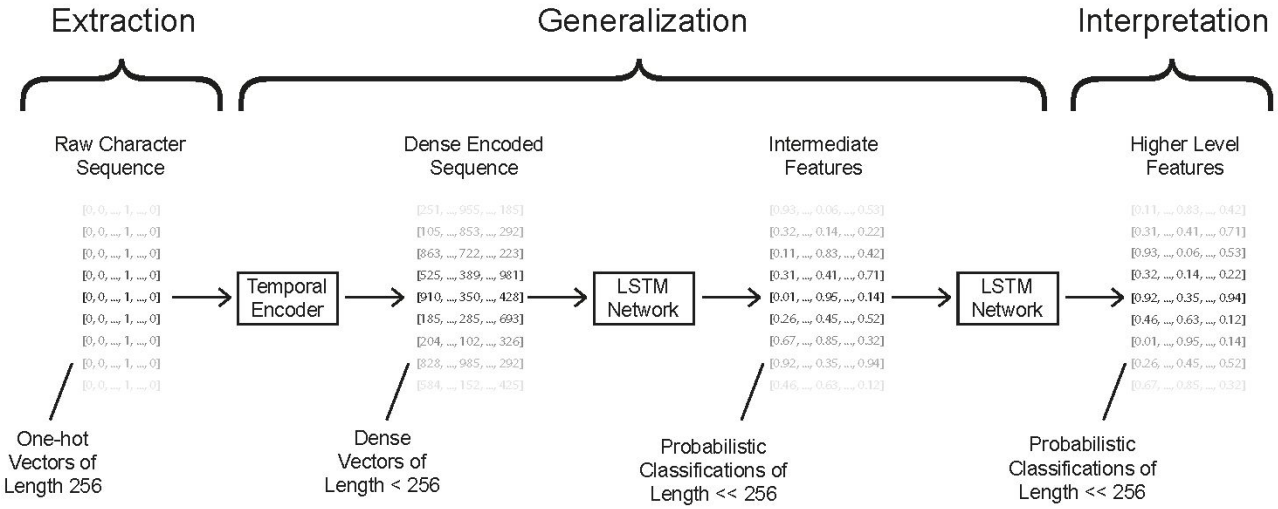
5.3 Mathematical Model

Sequences are stored as a three dimensional matrix as follows:

- The first dimension describes individual sequences.
- The second dimension describes time steps of a given sequence.
- The third dimension is a vector of features in each timestep



Above shows the matrix format in which sequences are stored.



Above shows the processing involved for a given sequence. Here, time is represented vertically and the sequence is fed to the network in a sequential fashion. The Long Short Term Memory recurrent network architecture has internal memory that is able to extrapolate from features given in previous timesteps. During preliminary tests, we trained directly on article quality, neglecting to train on intermediate features (known structures of the text such as links and templates). This led to poor results because there isn't much fundamentally different in structure between good and poorly written articles.

6. Plan of Work

Original Project Proposal

#	Names	Task
1	Luke Wielgus and Brian Faure	Data Processing, Visualization
2	Nathan Kjer and Wayne Sun	Data Classification, Analysis, Visualization
3	Adam Massoud and Brian Chu	Transforming, Visualization

Updated Project Proposal

Data Models (Nathan, Brian C.): Predict the classifications of a given text sequence

Database Parsing (Brian F., Wayne): Allow for the functionality of parsing through the data of wikipedia articles

Website (Luke): Create a basic webpage to allow for a basic user interface of the program

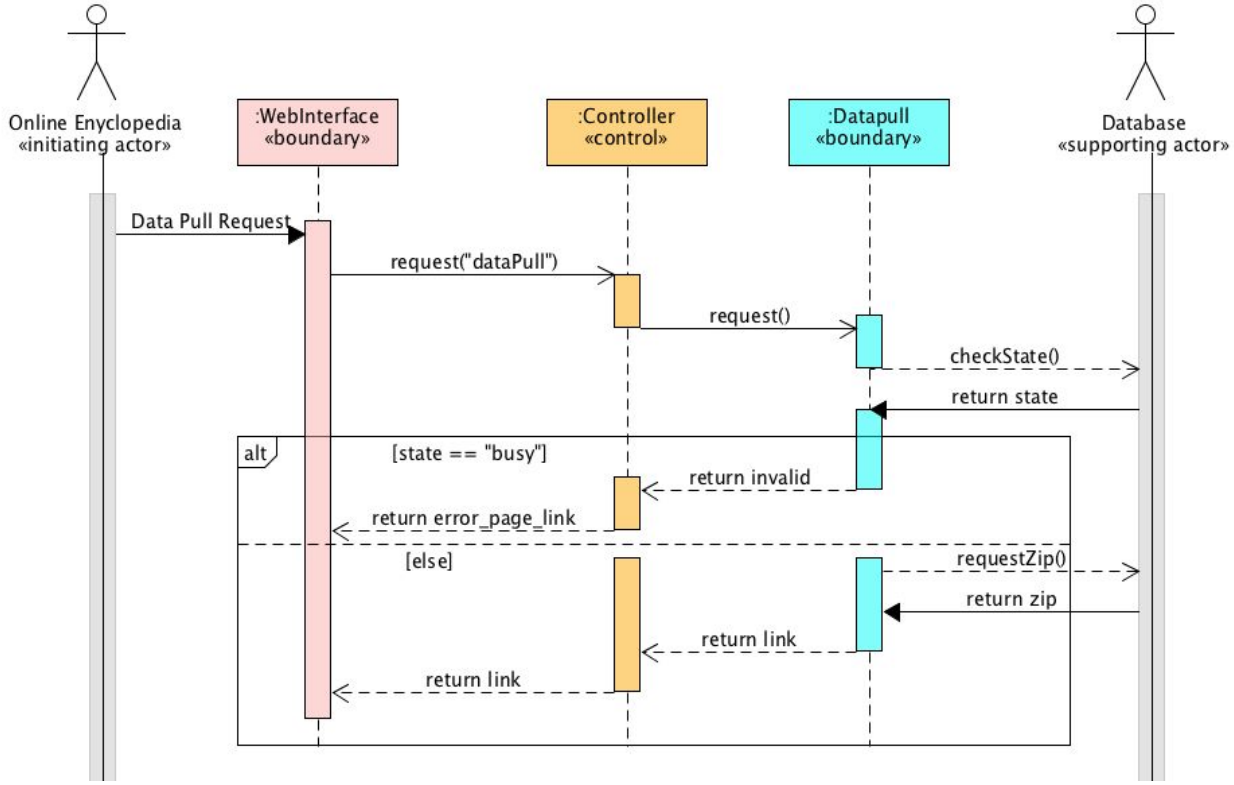
Chrome Extension (Adam, Deeraj): Create a chrome extension to extend the functionality of the website

7. Interaction Diagrams

UC-1 DataPull:

Interaction Diagrams

-Sequence Diagram

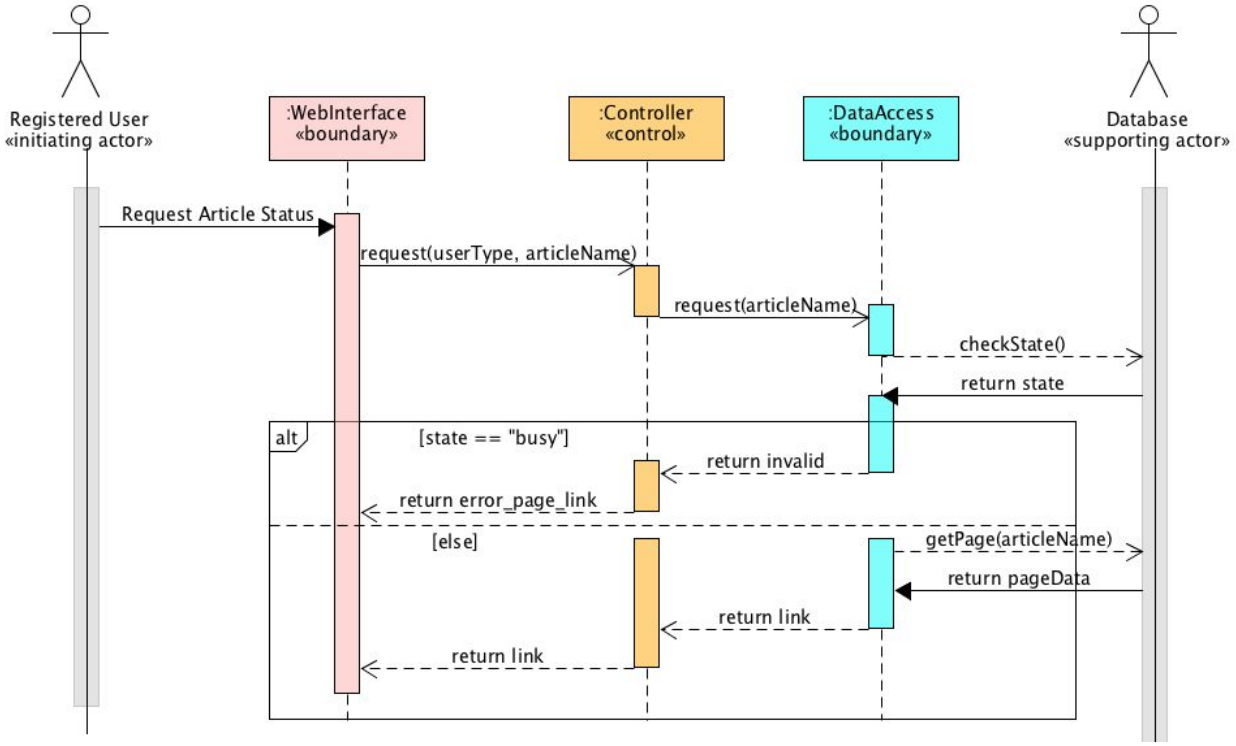


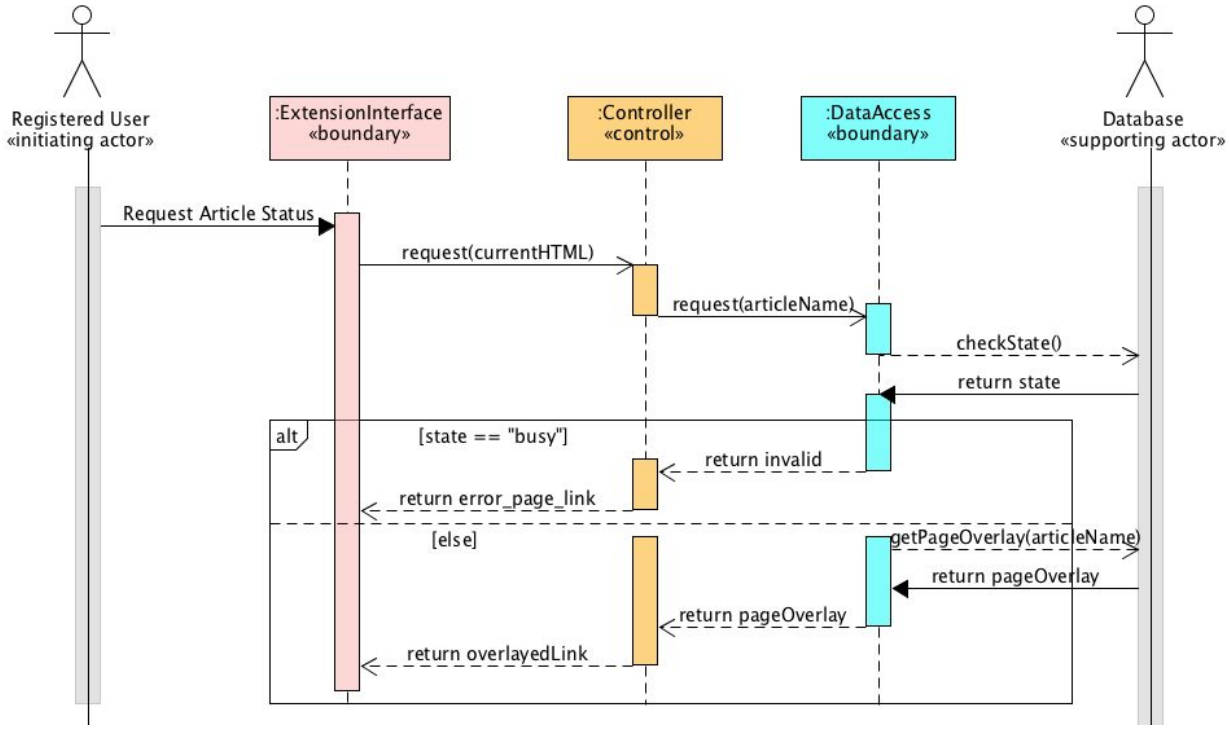
We used several common design principles in the creation of this process. For one, all interaction among the instances is limited to very simple instructions and expected responses, this will allow for room when designing for scenarios differing from the main success scenario outlined above. For instance, when the instance of Datapull is requesting a .zip file, we will be able to catch and contain the problem which arises if the Database is unable to return the link at that time (for updating/maintenance reasons etc.). In this case we can return a message to the Controller informing it that there was an error in pulling the zip file and Controller can direct the WebInterface to a webpage explaining the reason for the error.

UC-2 DataAccess:

Interaction Diagrams

-Sequence Diagram (Website & Chrome Extension)



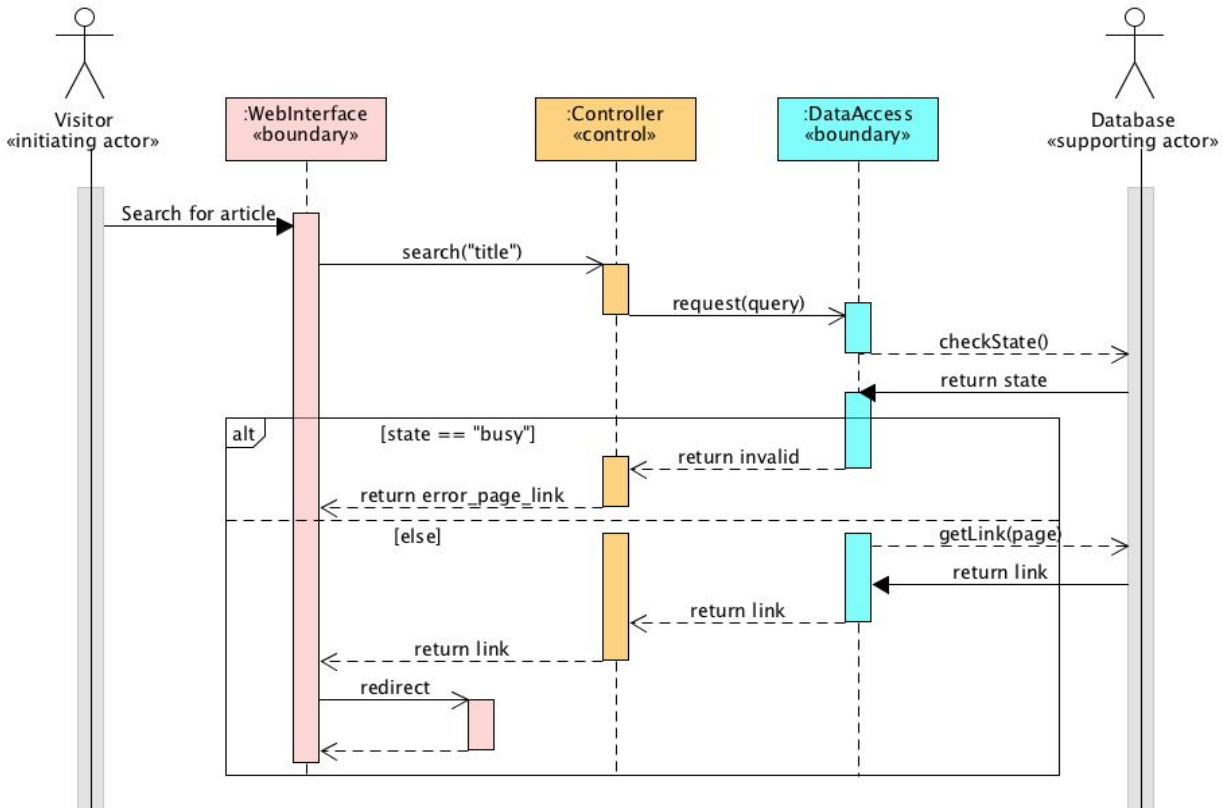


For this scenario, here were certainly a decent number of design principles that we had to consider. Much like how our Datapull use case works, we will use very simple and brief instructions between the interactions of the user and the Database for this use case in order to accommodate for the same scenario we addressed, which is any result not resulting in a success result, as illustrated above. As far as we know, how the process should work is that the user will come to the website or access the chrome extension and search for some topic, thus signalling to the controller that a request for information has been made. This then simply means that our controller will ask the DataBase for this information, which then is simply given back to the controller once the information is pulled. Finally, now that the information is given, the controller will then simply display this information to the user. Now, obviously this is not going to happen all the time, so we have to account for a “worst case”, or alternate, scenario as well. For example, if the Database is busy at the moment, due to an update being made or maintenance being held at the time the user asks for his/her request, the controller will then simply send an “error page link” screen back to him/her with a description that the system is currently busy at the moment. Therefore, this both satiates the database to allow itself to update peacefully or be fixed without any interruptions while also successfully telling the user that there was an error while he/she tried to access data and tell him/her why that occurred, all being done in a timely and efficient manner due to the simplicity of our design.

UC-5 VisAccess:

Interaction Diagrams

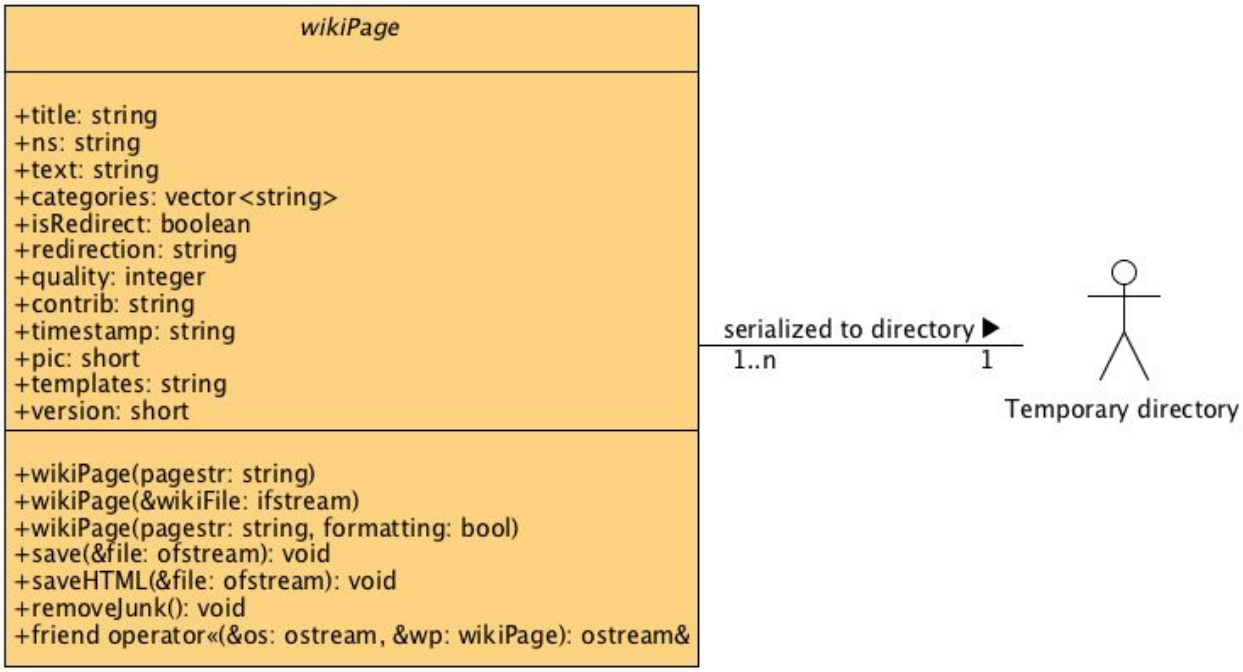
-Sequence Diagrams



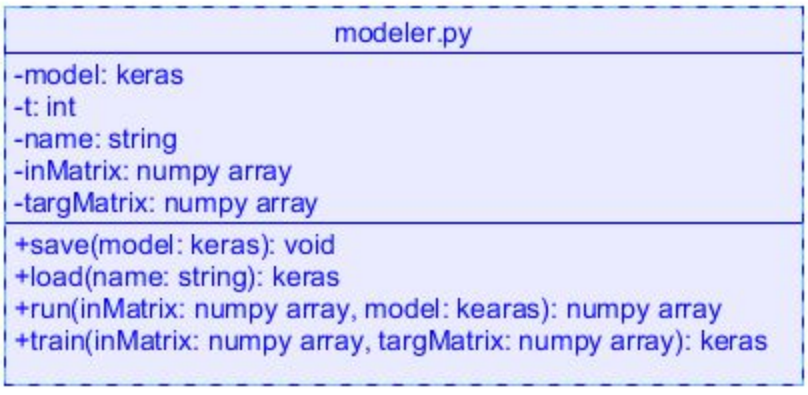
For our VisAccess process, we can obviously see the similarities between this and our DataAccess scenario. Now, the only difference would be the fact that any DataAccess scenario can begin either from the user entering the website or using the chrome extension to access the information required from the site. The scenario, however, does in fact go about the same steps, and therefore, it is just as straightforward. The success scenario is displayed above and highlights how similar the two processes are to one another while also acknowledging the same “worst case” or alternate scenario where the Database is busy while the visitor makes his/her request because this can happen no matter who is searching for something on the site.

8. Class Diagram and Interface Specification

8.1 Class Diagram



→ wikipedia.cpp takes reads the official wikipedia data dump (xml format) and parses it into millions of wikiPage objects. This is done using the wikiPage(pagestr: string, formatting: bool) function which is provided with small portions of the xml file at a time in string format. The constructor is also provided with a boolean which describes whether or not to parse out all of the formatting from the original xml articles. After the wikiPages have been created, they are saved as either plaintext or HTML format or both within the 'Temporary Directory' I have designated above. At this point the Python script can draw from the 'Temporary Directory' for its neural network implementation.

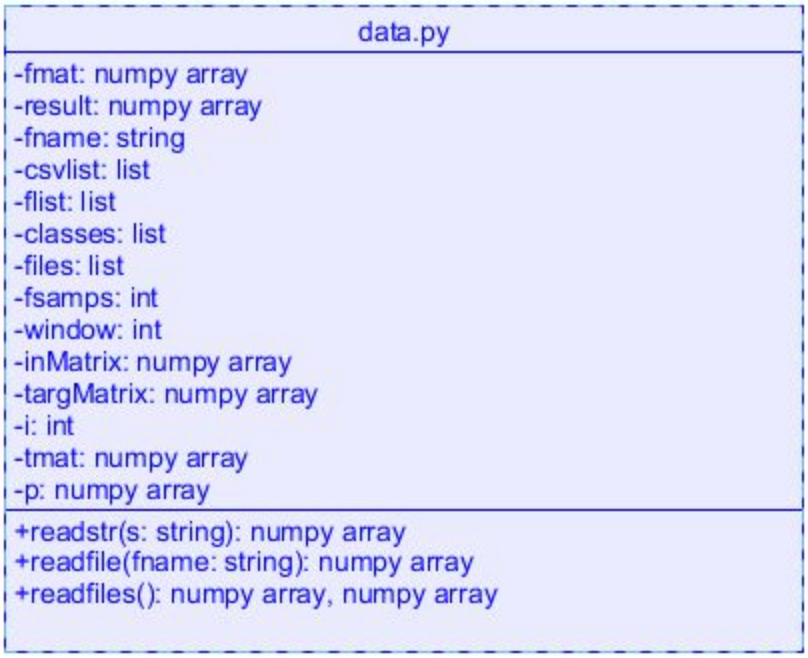


Methods:

- Train learns a model for the relationship between its arguments: an input numpy array and a target numpy array.
- Once this model is created and the error minimized, the model is saved using the save function.
- To load a model, the load function is used.
- Run is a wrapper function that runs a given model given an input matrix.

Attributes:

- Model: a keras model; keras is the machine learning library for the recurrent network infrastructure.
- inMatrix: The input data, in sequential matrix form
- targMatrix: The desired output of each sample from the inMatrix



Methods:

- Readstr converts a string into a sequential numpy array.
- Readfile reads a given file and converts the string into a sequential numpy array.
- Readfiles performs readfile on multiple files, specified in a csv.

Attributes:

- fmat: a matrix of a given file, where each element is a decoded byte
- csvlist: a list of lists, taken from a csv file
- flist: a list of files, to be read
- inMatrix: The input data, in sequential matrix form
- targMatrix: The desired output of each sample from the inMatrix
- window: how long each sequence is

8.2 Data Types and Operation Signatures

wikiPage Data Members:

+title: string → The title of the Wikipedia article (from data dump)
+ns: string → The namespace of the Wikipedia article (from data dump)
+text: string → The body of the Wikipedia article (from data dump)
+categories: vector<string>
→ Includes the categories the Wikipedia articles resides in (taken from data dump)
+isRedirect: boolean
→ True if the Wikipedia article redirects to another article instead of being its own page
+redirection: string
→ If the page is a redirect, this contains the name of the redirection destination
+quality: integer
→ 0=redirect, 1=regular, 2=good, 3=bad, quality of the page
+contrib: string
→ The name of the most recent page contributor
+timestamp: string
→ The timestamp of the most recent page edit
+pic: short → The number of pictures in the article
+templates: string
→ The formatting templates found in article (things pointed out by other Wikipedia users that should be changed)
+version: short
→ Corresponds to the save version number, used internally only in some situations (i.e. when recovering wikiPages from a directory)

wikiPage Operations:

+wikiPage(pagestr: string)
→ Constructor which takes in a string that contains a single wikipedia article including all of the formatting. It sets all of the data members from information it finds in the string

+wikiPage(&wikiFile: ifstream)
→ Constructor which takes in a file containing a single saved wikiPage using the save version 1.0 and turns it into a new wikiPage (i.e. deserializes a wikiPage)

+wikiPage(pagestr: string, formatting: bool)
→ Constructor same as the first one except has option to leave all the formatting artifacts from the data dump text

+save(&file: ofstream): void
→ Takes in a reference to a ofstream file and saves the wikiPage in save version 1.0 (see plaintext in 9.4)

+saveHTML(&file: ofstream): void
→ Takes in a reference to a ofstream file and saves the wikiPage in HTML version (see HTML in 9.4)

+removeJunk(): void
→ Only called in some circumstances, removes all but the plain alphabet from the text of the wikiPage (body of Wikipedia article)

+friend operator<<(&os: ostream, &wp: wikiPage): ostream&
→ Overloaded << operator to output the wikiPage in custom format; outputs all wikiPage metadata.

8.3 Traceability Matrix

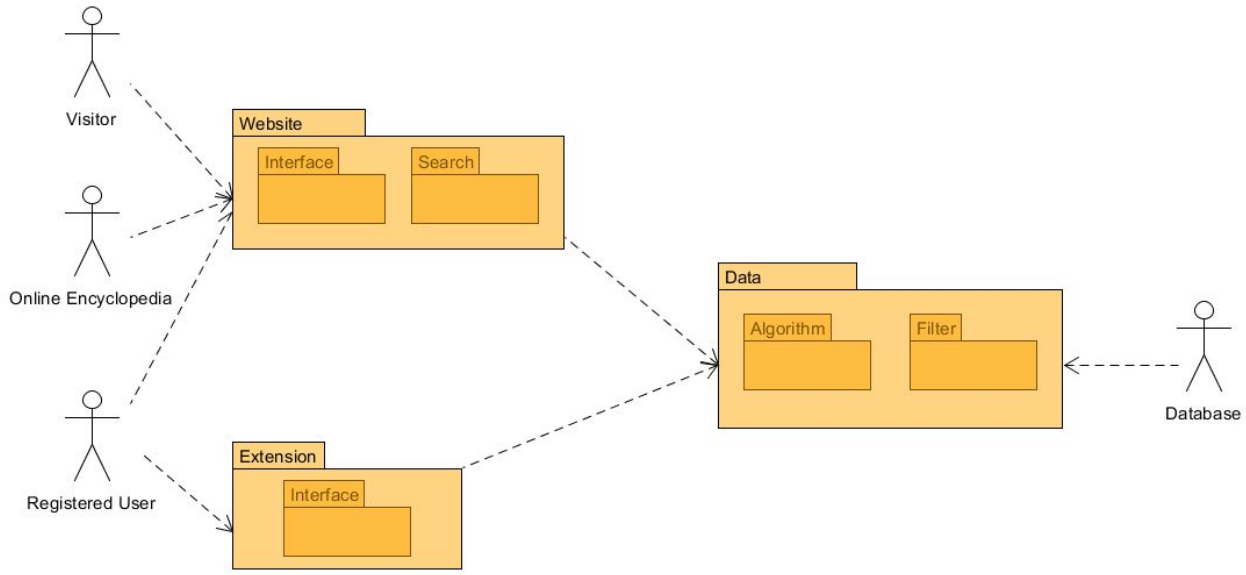
Domain Concepts	Classes		
	wikiPage	**modeler.py**	**data.py**
Controller			
DataPull	DataPull will assemble a .zip compressed file of the entire library of wikiPage ratings, pulled from the wikiPages themselves.		
DataAccess	DataAccess will allow the registered user to view details about wikipedia articles which are assembled to html using the saveHTML member of the wikiPage class.		
LoginVerify			
ReportHandler			
UserRegistrar			
ExtensionInterface	ExtensionInterface will pull the metadata of the wikiPage (some data members not yet added) to create the text overlay on the user's web browser.		
WebInterface			
PostProcessor			
UserSearch	The input gathered when a user searches will be tabulated against the article titles in the wikiPage metadata.		
LoginStorage			
ModelTrain		Uses data obtained from data.py, in matrix form, to learn categorical relationships.	Extracts the html data saved by wikiPage to be classified.
ModelRun		Uses a trained model, perhaps after loading a saved model, new text is classified and labeled using data.py	Interfaces with the html upon classification results for display.

9. System Architecture and System Design

9.1 Architectural Styles

- ***Client/Server***: The way WikiClassify is set up is such that users (clients) will be searching for their desired Wikipedia article. These articles will be stored in a database (server) and when one is asked for by the user, the search will be displayed back for them. This is done through a website or extension; the user will search for the article on one of the specified interfaces, and the desired article will be retrieved from the database and returned to the user.
- ***Object Oriented***: Due to the fact that our design incorporates many different components, we have found it necessary to split our code among different modules, each of which performs its own task. For example, we have a c++ file which fetches data from the Wikipedia data dump xml file, parsing it into wikiPage objects, and then saves it as both html formatted files and plaintext Wikipedia articles. We then have a python file which can take these wikiPage objects and run them through our neural network, completely cut off from the c++ file. By keeping these files separate and only interacting through saved documents we can create a coding hierarchy which is easy to explain and even easier to edit.
- **Message Bus**: Our system is designed to take in search parameters from a user and send them the corresponding data without the need to of using specific details.

9.2 Identifying Subsystems



The subsystem maps lay out the three main large subsystems that form our system. The Extension subsystem handles Registered Users who are requesting information on a specific article they are browsing. The Extension subsystem will handle processes relevant to extension user including user verification,information retrieval and display of modified article on the output. The Website subsystem handles all users that come to search articles. It has processes to take search queries and send proper requests to database for parsing. The Website subsystem also includes the Interface for the users to input parameters and view output results from the Database. The Data subsystem processes requests from users and queries the database for the desired data. Relevant information such as article names,search parameters,target users,database status would be processed by the database subsystem to ensure proper data from the database is sent to the correct user. The Data subsystem will also handle status alerts including blocking access to the database during updates or downtime.

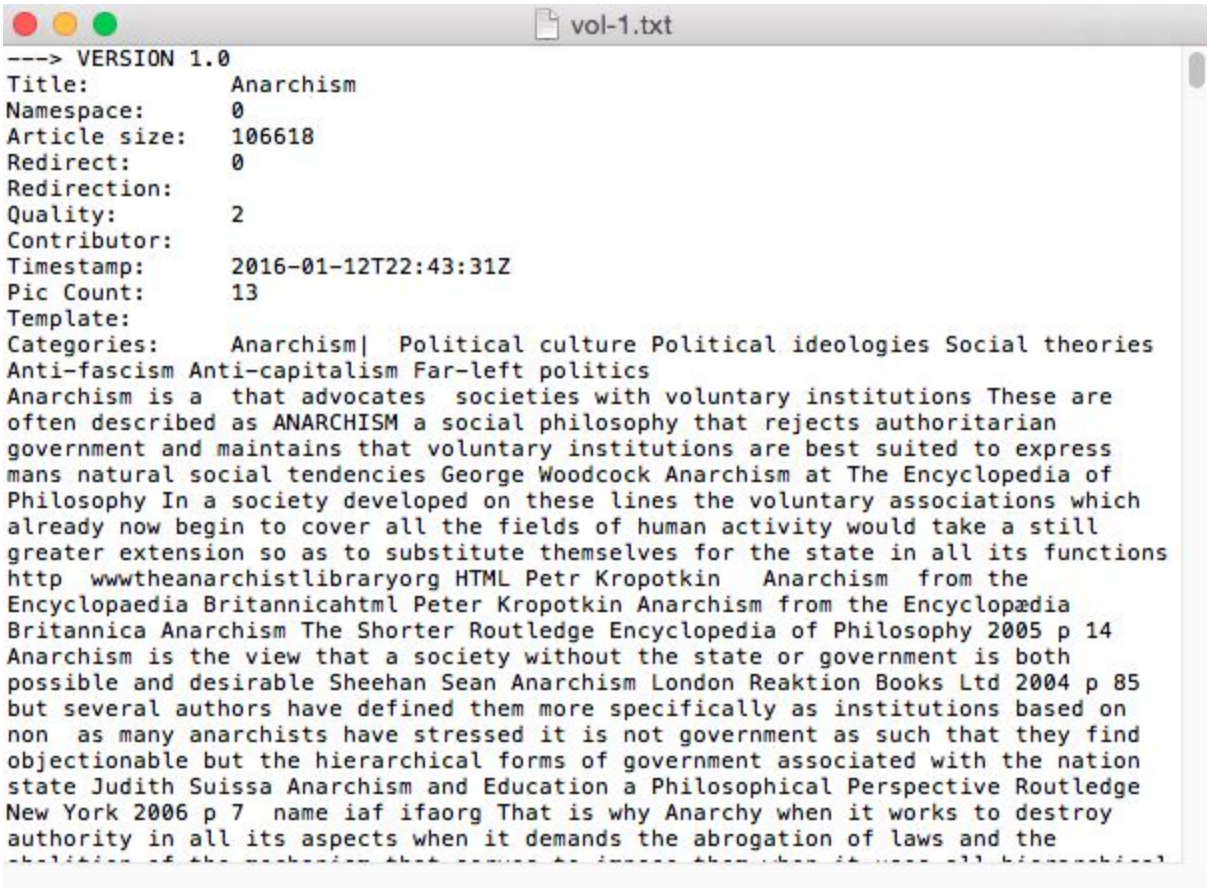
9.3 Mapping Subsystems to Hardware

All the data (Wikipedia articles, usernames/passwords, etc.) will be stored on a single computer, which we will consider our server.

9.4 Persistent Data Storage

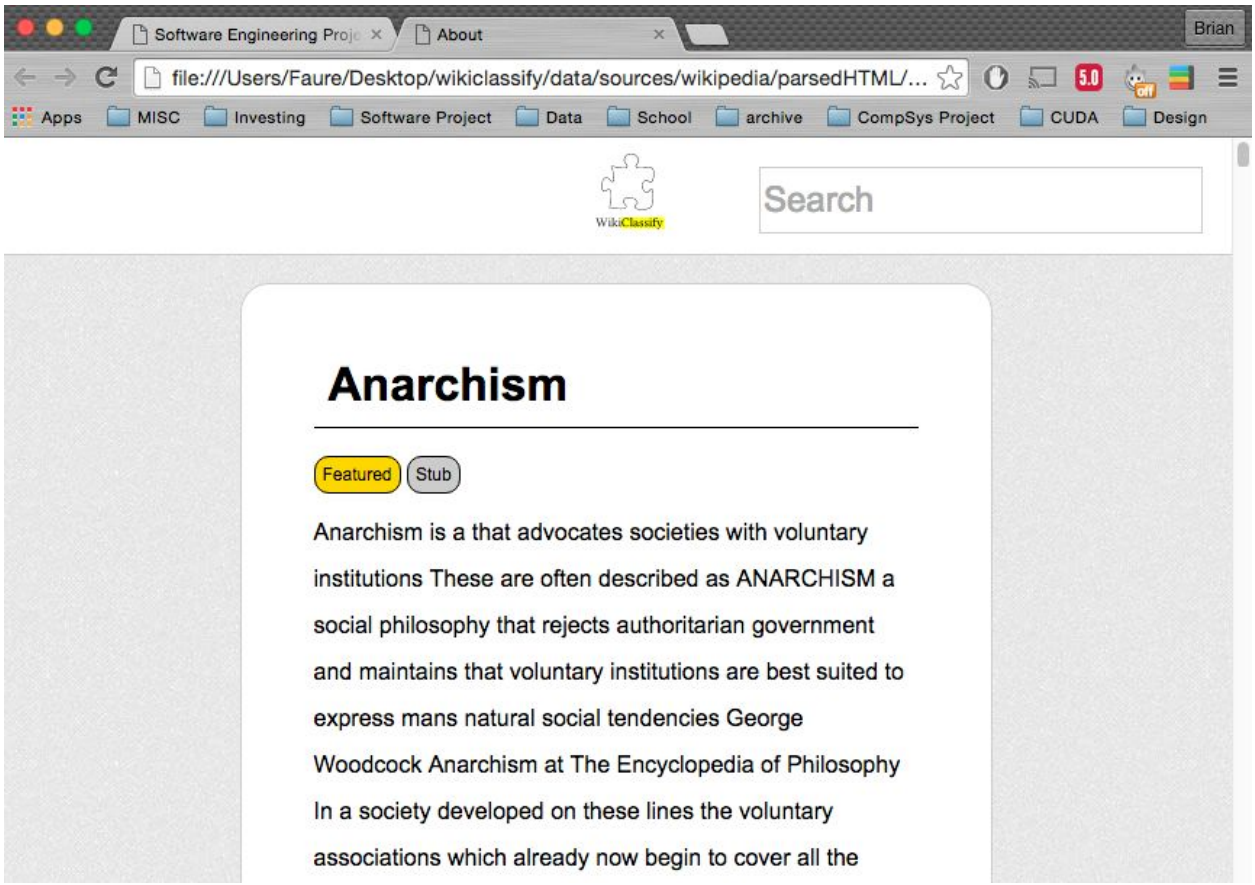
Our system will need to save data that can outlive a single execution, as there will be multiple users with multiple search queries. The data is currently stored as .html and .txt files, and thus our storage management strategy is that of a flat file database.

Plaintext File Type:



As seen by the picture above, the plain text files are simply the Wikipedia articles in plain text format. All the formatting has been parsed out. Furthermore, the article details have been extracted and placed at the top of the file.

HTML File Type:



As seen by the picture above, the .html files contain the head and body of the Wikipedia articles. As we are still in the early stages of implementation, the .html files do not have the text overlay/highlighting that the final product will have.

9.5 Network Protocol

The system will being using a Nginix web server as a modern alternative to apache. It is a modernized web server for use with HTTP. Nginix was chosen as it is a web server with better thread control and security vs apache. Apache is the most popular web server but is old and is subject to greater chances of security breaches being brought to light due to popular. The web page access methodology allows users to get desired data without having to download complicated software or large amounts of data.

9.6 Global Control Flow

The system is procedure-driven in a linear fashion. This is because users will always have to go through either the website or extension, and then search for the article they want. After searching, the system will always follow the same steps; look for the article in the database, and return the article to the user. There is no time dependency in our system. Our system will use multiple threads in that at any given time there could be multiple users searching for articles. However, there will not be any separate threads of control within each individual user's case. The system itself will internally handle multiple thread data synchronization for accessing the classification data.

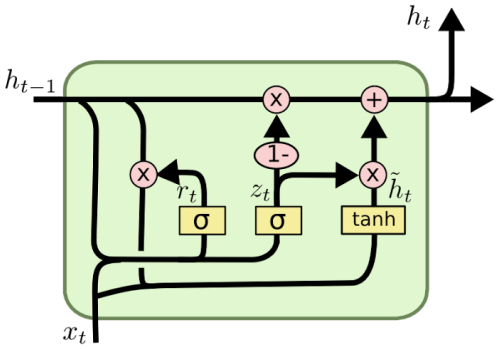
9.7 Hardware Requirements

One computer will be used as our server to store the data of the classifications and the user login information. This computer will need approximately 150 Gbytes of hard disk space to ensure we have more than enough storage for operation. This will hold the two most recent Wikipedia dumps, the current one and one that is a month behind as a backup. The server will also need a minimum network bandwidth of approximately 1 Mbps to allow for the data transfer. The server should have a modern dual core processor or better in order to handle data processing requirements for the server to parse and respond to requests in real time for various numbers of users.

10. Algorithms and Data Structure

10.1 Algorithms

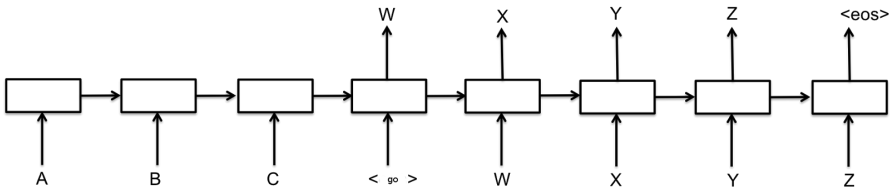
Long Short Term Memory



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Credit: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Given an input vector x_t at time t, and hidden state h_{t-1} at time t-1, the current hidden state (and resulting output state) at time t is computed using the equations above. These structures can be fed into another and stacked in layers to learn greater complexity. These memory cell architectures are much more robust to recurrent (temporal) tasks than simple neural networks. Libraries which take advantage of symbolic gradient calculation, matrix optimizations, and GPU support are preferable over hand-coded implementations.



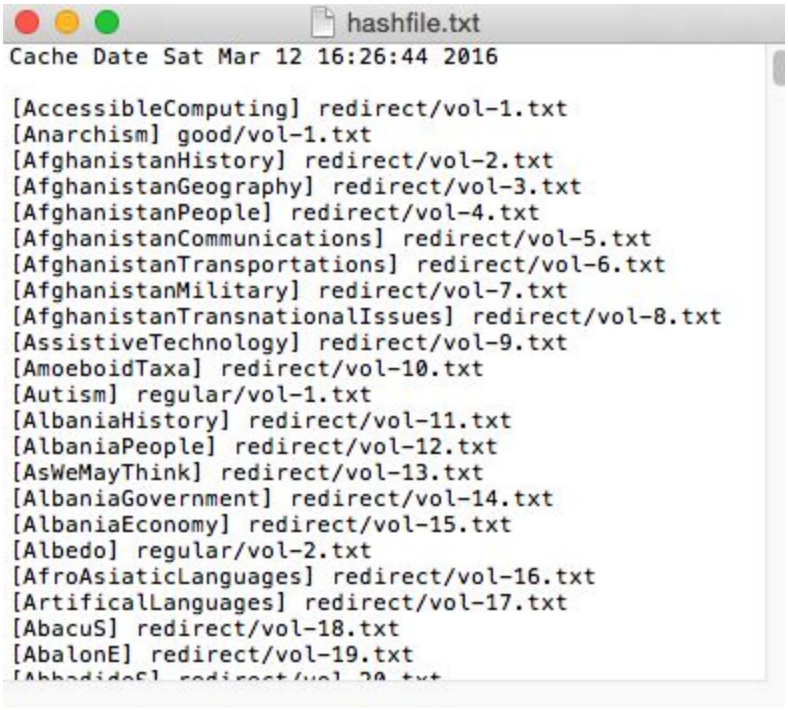
Credit: https://www.tensorflow.org/versions/r0.7/tutorials/seq2seq/index.html

10.2 Data Structures

In the data parsing file (wikipedia.cpp), the data dump xml file is parsed into millions of individual .txt and .html files pertaining to each article. These article files are placed into 4 individual folders while their names are placed in a hash table along with their relative folder locations. This hash table is saved as a .txt file and allows for much faster access and searching by allowing the algorithm to simply check each line of the file for a title match then following the relative path to find the final destination. This is opposed to having to search

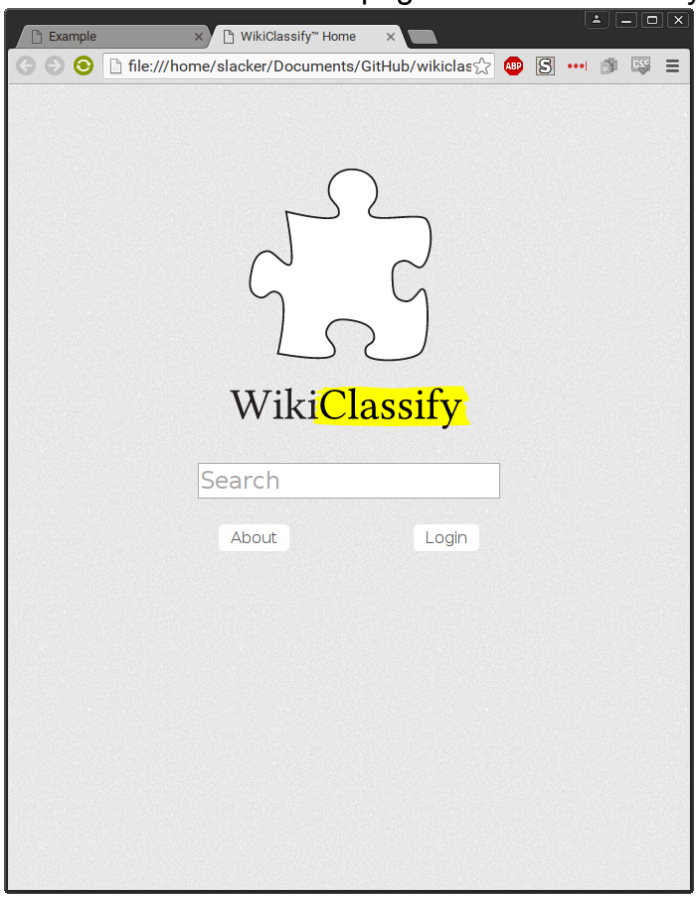
through each folder and trying to match the titles from there. In the future we hope to find an even more efficient form of searching and sorting the database.

Hash table text file:

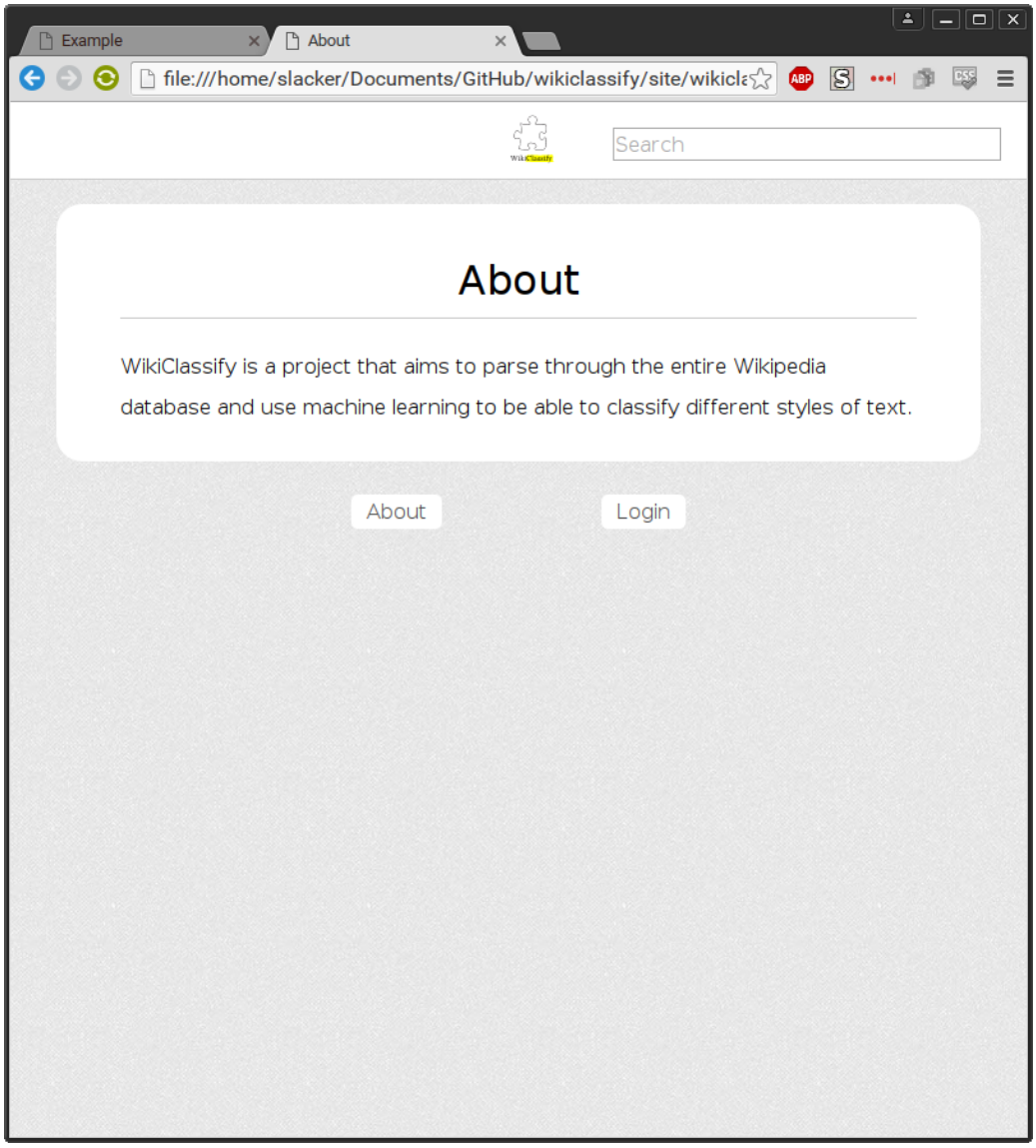


11. User Interface Design and Implementation

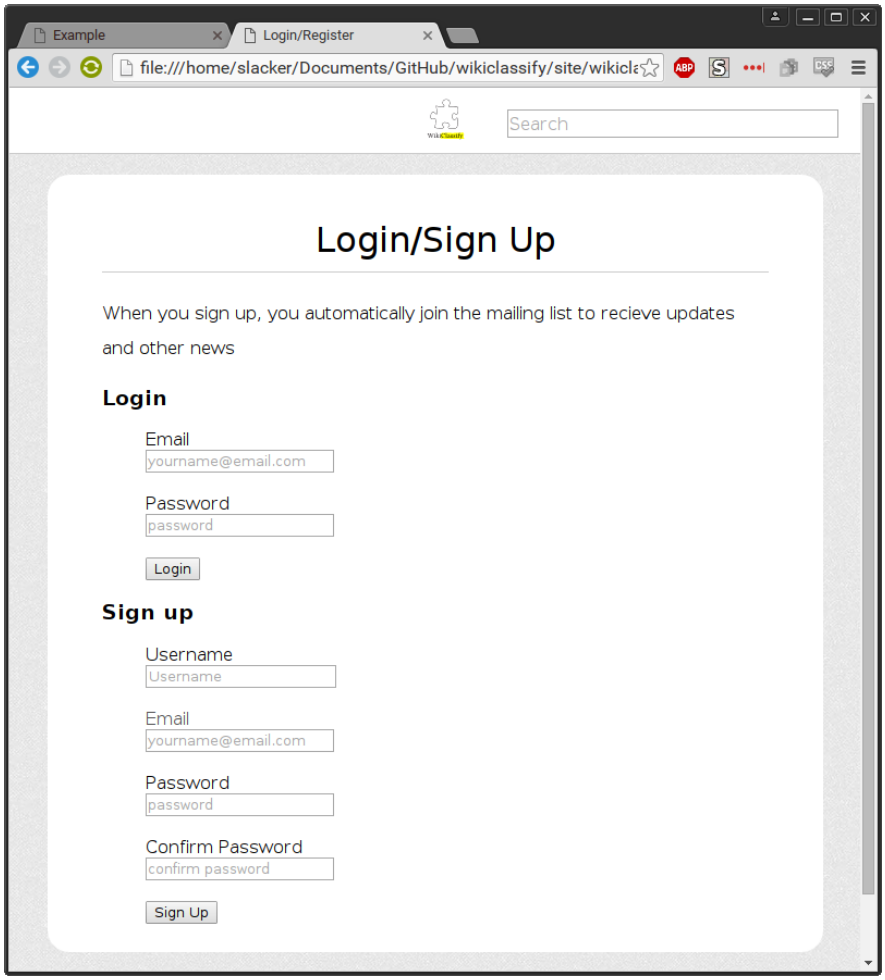
Current User Interface:
Home:
Here is the current home page of the WikiClassify site/UI



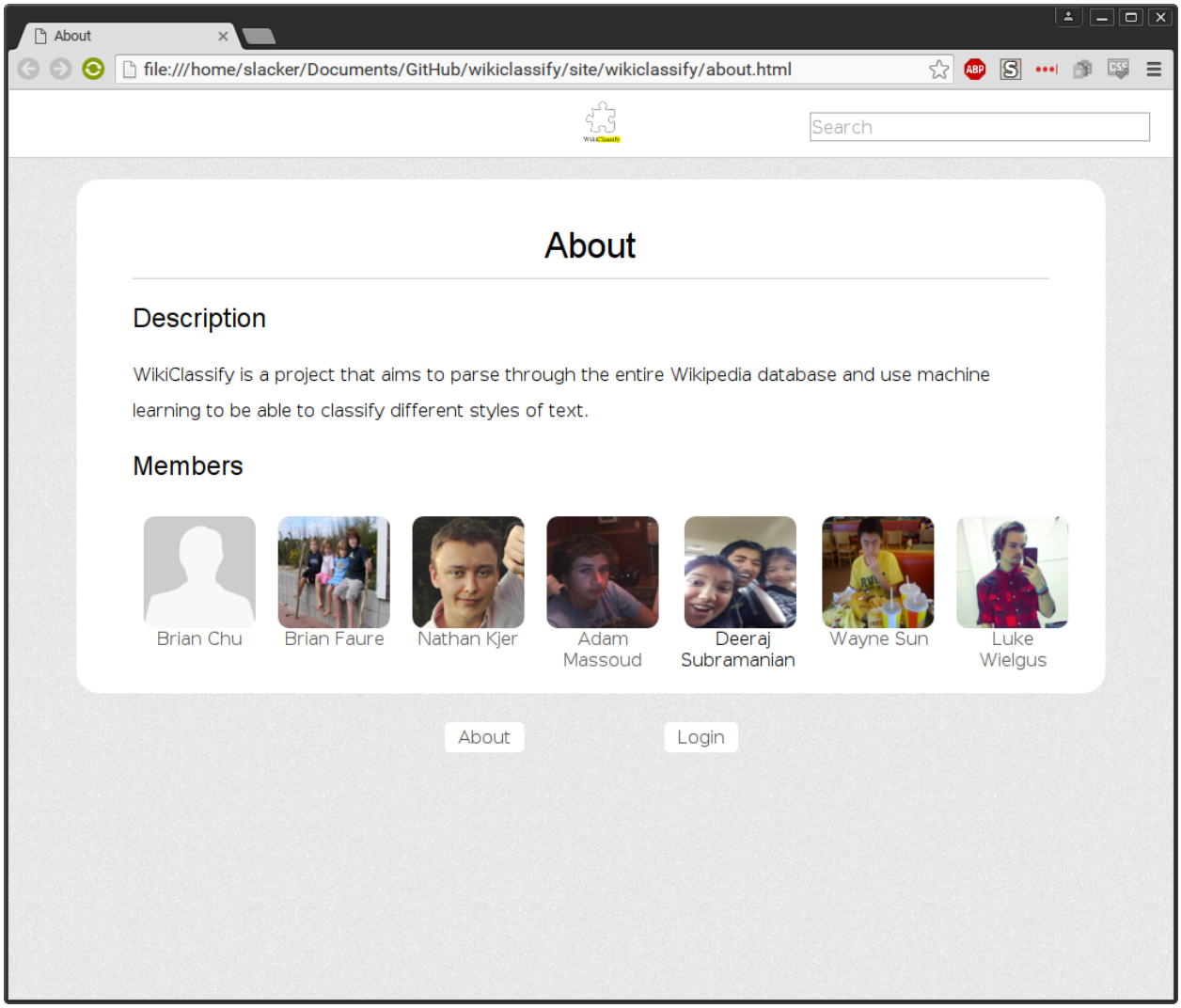
About:
This is just a basic page that explains what the project is about


Example
About
file:///home/slacker/Documents/GitHub/wikiclassify/site/wikicla
Search
About
WikiClassify is a project that aims to parse through the entire Wikipedia database and use machine learning to be able to classify different styles of text.
About
Login

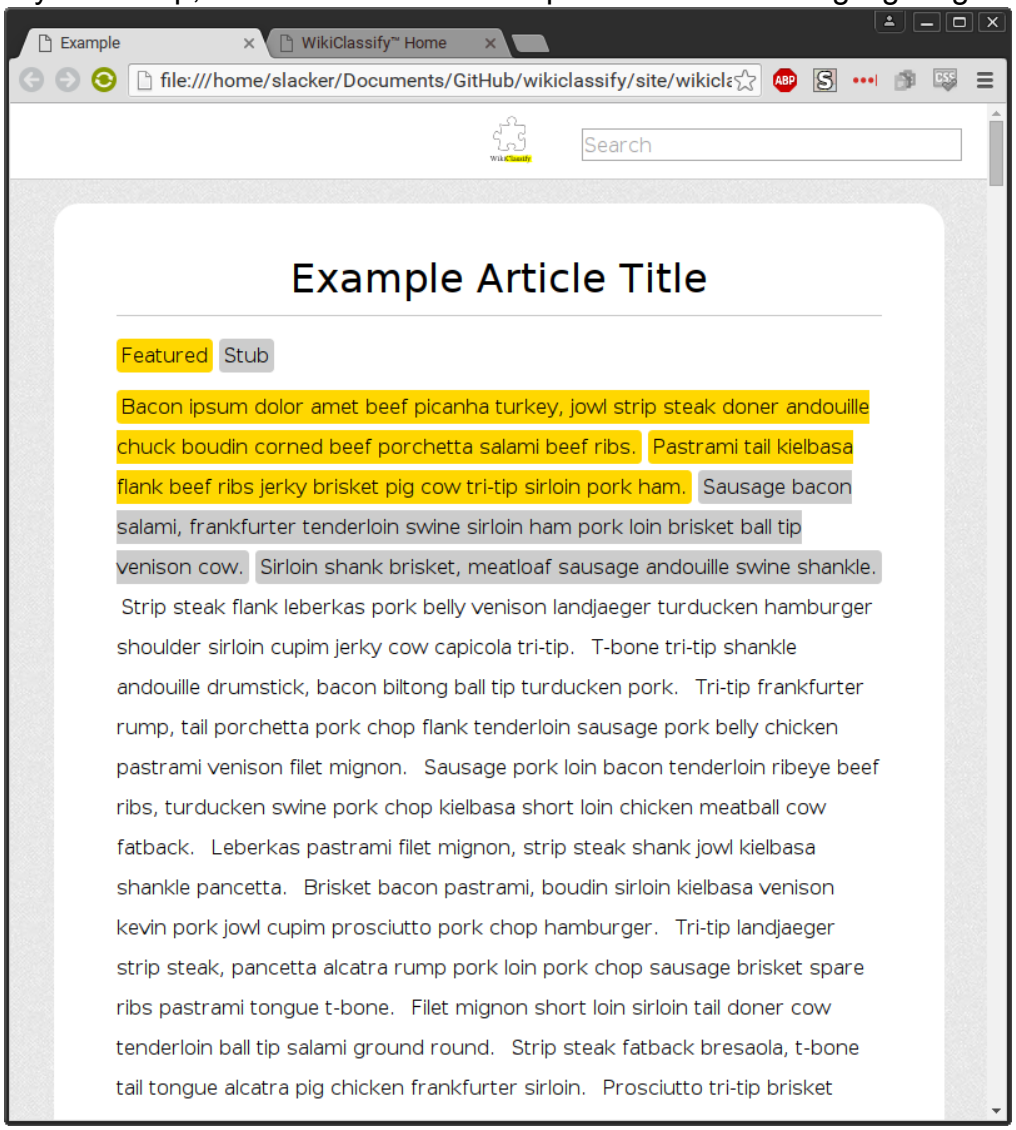
Login:
This part of the UI allows the user to sign-up and login to the website.



About:
This page gives the user information on our product as a whole.


About
file:///home/slacker/Documents/GitHub/wikiclassify/site/wikiclassify/about.html
Search
About
Description
WikiClassify is a project that aims to parse through the entire Wikipedia database and use machine learning to be able to classify different styles of text.
Members
Brian Chu
Brian Faure
Nathan Kjer
Adam Massoud
Deeraj Subramanian
Wayne Sun
Luke Wielgus
About
Login

Example:
The example below shows what the webpage UI will look like after a user searches an article. The things that are tagged will be highlighted in different colors and shown with a key at the top, and the article will be reproduced with the highlighting below that.



12. Design of Tests

Possible Tests:

-**Test for chrome extension**

-Coverage for chrome extension: successful case, user connection stall,database connection stall,or maintenance, extension errors(the user requests and extension sends bad request or issues on user end with webpage directly)

-Unit Tests will make sure parsing module and machine learning module can be used together

-**Test for data Access use case**

Coverage for data access: successful cases,connection stalls during retrieval process,sudden faults(someone accidentally unplugs the server power cord),corrupt data/nonexistent data

-Unit Tests for modules involving user input and accessing the corresponding data. Unit tests will also ensure basic outputs are working as prelude to integration testing.

-**Test for user reported error use case**

Coverage for user report: connection stalls,gibberish from user

-Unit Tests for pull requests to database, general data access(before we check for user), web Page and extension in general(make it work)

Our integration testing will involve making sure that the website/extension (user interface) can be integrated successfully with the machine learning software and its results. In other words, the integration testing should guarantee a working model of the entire system, regardless of the completeness of the aesthetics or anything not involving implementation of the use case. This will involve the unit tests and will allow for a smooth transition to the validation testing.Integration testing will also ensure compatibility between parts such as data access receiving requests from UI and returning proper information for UI to display. Integration test will be composed of two parts. The first part will be data base and parser module connection. This phase will consist of making sure data access and the parse process goes smoothly including access to final results,updating the database and returning results. The system here will be feed example data and send dummy requests to ensure the results are both proper and being sent correctly.The second phase will be to integrate the results of the first phase to both the web page and chrome extension to create a fully working system.

13. Project Management and Plan of Work

13.1 Merging the Contributions from Individual Team Members

The majority of merging will be handled with the help of github and running backend sever.

13.2 Project Coordination and Progress Report

Currently the parser module is fully functional i.e. it can take the input (datadump xml file) and transform that into a directory of wikipedia articles. There is still some finalization required in making the module more efficient and increasing its compatibility with the rest of the modules.

The webpage is also made with basic functionality. It has a login/register page but does not yet store user login info. It also displays properly with pre-picked articles to use as examples.

13.3 Plan of Work

Once we have finalized the parser and the machine learning algorithm we will have all of the background finished. At that point we need to create the interface between the website (and extension) and the database of categorized wikipedia articles. This will most likely consist of some sort of server we will set up along with the php embedded on the website to communicate with the database and manage connection between the user and the database. As I can see it, the most important use case will be dataAccess where the user enters the title of an article they wish to search for and our php will translate this into some sort of searching algorithm which traverses through our database to find the correct result. This use case will most likely also be used as the connection between the chrome extension and the database but will we have other issues to address in that context as well. For example, when the user has installed the chrome extension and is on a Wikipedia article page, will the chrome extension send a copy of the users html to the database and return a version of the html with our text highlighting feature added or will the chrome extension just redirect the user to our website entry of the article they are browsing on Wikipedia.

13.4 Breakdown of Responsibilities

Brian Faure → Parsing Module…

-Processes the .xml data dump file

-Creates database to interface with the machine learning algorithm

-Pulls important data from the .xml along with splitting file along articles

-TO DO: Implement a more efficient file structure to increase indexing and Sorting efficiency when handling post-parsed articles

Luke Wielgus → HTML/PHP webpage

-Display a search bar

-Allow user to search a given article

-Display the text from the wiki-dump

-TO DO: login data storage

Nathan Kjer → Machine Learning Processing

-Read and interpret parsed Wikipedia dump

-Write interpretation back to text for web display

Wayne Sun → Integration of Machine Learning and Software Pipeline

-Allow machine learning software to work on user interface

-Database updates

Brian Chu →Assistance with Machine Learning Processing and integration between UI front end and software pipeline.

References:

---

1. https://computation.llnl.gov/casc/sapphire/overview/overview.html
2. http://infojustice.org/wp-content/uploads/2013/10/band-gerafi10032013.pdf
3. http://digitalcommons.law.yale.edu/cgi/viewcontent.cgi?article=1055&context=yjolt
4. https://en.wikipedia.org/wiki/Wikipedia:List_of_bad_article_ideas
5. https://en.wikipedia.org/wiki/Reliability_of_Wikipedia
6. https://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria
7. https://en.wikipedia.org/wiki/Wikipedia:Good_article_criteria
8. https://en.wikipedia.org/wiki/Wikipedia:Maintenance

Pictures:

https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg
http://scikit-learn.sourceforge.net/0.5/modules/clustering.html
https://commons.wikimedia.org/wiki/File:Long_Short_Term_Memory.png