Spring 2010

Measuring Balkanization in Wikipedia

Colin Welch
Macalester College

Honors Project

Macalester College

Spring 2010

Title: Measuring Balkanization in Wikipedia

Author: Colin Welch

Measuring Balkanization in Wikipedia

Colin Welch

April 22, 2010

Macalester College

First Reader: Shilad Sen

Second Reader: Libby Shoop

Third Reader: Andrew Beveridge

Abstract

Modern society has become increasingly *balkanized*, or ideologically polarized and socially fragmented. Political parties interact with each other over divisive issues by using polarizing rhetoric. Internet users create small, opinionated communities like dKosapedia and Conservapedia, Wikipedia-like websites written from positions of left-leaning and right-leaning bias. Even Wikipedia may not be immune to balkanization. As a free encyclopedia written by users from an unbiased point of view, it is in the interest of the general public to keep Wikipedia as free of balkanization and polarization as possible. If Wikipedia authors are free to express their conflicting points of view, the quality of information may degrade and the community could become divided. This might lead to less time being invested in improving articles, and more time spent resolving conflict. Moreover, balkanization could also lead to users developing discipline-specific editing norms, creating less cohesiveness in the overall site.

Wikipedia offers a chance to investigate balkanization on a large scale, but is also difficult to study for exactly this reason. Wikipedia represents terabytes of text. The restrictive size of the dataset and the lack of available software libraries for parsing make working with Wikipedia difficult.

This thesis presents a software library capable of handling Wikipedia's large dataset by using parallel processing techniques. The software library parses Wikipedia revision histories into a graph model so that we can investigate balkanization on Wikipedia using established graph theory metrics. I contribute both the Java software library for parsing revision histories and implementations and analysis of three possible balkanization metrics: density, degree centrality, and conditional probability.

Table of Contents

List of Figures

1. Introduction

In the recent health care debate in America, individual factions grew more radical as they talked only amongst themselves. As the bill itself went from a strongly liberal bill to a much more conservative one, each political side experienced polarization. The right went from opposing the public option to opposing any health care reform. The left shifted from putting forth the original strongly liberal plan to barely supporting the final bill. Calls to "kill the bill" came from both ends of the spectrum [4, 9, 11].

These polarized conflicts can be seen on Internet sites as well. Because of the unbiased nature of information available on Wikipedia [30], Internet users have established Conservapedia [8] and dKosapedia [19]. These are opinion-focused political encyclopedias written from positions of explicit bias, where users don't need to interact with opposing viewpoints at all.

The situation in the American health care debate and the separation of opinion on Conservapedia and dKosapedia are both examples of balkanization. Balkanization is the process of large groups of people subdividing along specific lines, often cultural, political, or religious ones [3]. This process helps create and reinforce social polarization [22]. Non-political disciplines are not immune from this process. For example, authors on Wikipedia with a specialty in molecular biology may be more likely to edit articles relating to biology than articles about the geography of the Ural Mountains. These biologists would also be unlikely to have a reason to have a conversation with users who specialize in articles about Ural. If these two groups of editors could interact regularly, they could develop strong inter-disciplinary relationships. These relationships could then lead to a more cohesive editing style and sense of community throughout the site.

Wikipedia is an unbiased resource freely available to millions of people, so it is important that it does not become overly balkanized. Polarization of author opinions in divisive areas like politics and religion can lead to situations like the use of blatant bias on Conservapedia and dKosapedia. Over-specialization of authorship can cause users to stop writing according to communal norms in favor of developing discipline-specific writing styles. This lowers the overall cohesiveness of the encyclopedia and attractiveness to new editors. Both of these effects hinder Wikipedia from maintaining its normal level of quality, which in turn influences information available to the general population.

Combating balkanization is complicated - there's no single answer to the problem of social polarization. However, being able to detect balkanization is a good first step. This research seeks to answer the question, "Can evidence of balkanization be found on Wikipedia?"

Wikipedia is structured as a collection of users, articles, and links between them. Links can vary from one article linking to another to one user mentioning another user on a third user's User: talk page. We map these relationships onto a graph that uses articles and users as vertices and links as edges. Using this model allows us to use graph theory metrics to test for balkanization.

The English language Wikipedia has 3.2 million articles, with 139 million total revisions spanning all articles. Some full revision histories are over 15 GB for a single article. The

sheer scale of the Wikipedia dataset makes conventional analysis techniques inadequate. To efficiently analyze this data we use parallel processing, specifically the Hadoop map/reduce library, as the base of our software library.

The main contribution of this project is a software library for parsing Wikipedia revision histories into a "link graph" of edges and vertices, which can be used in the future for more in-depth analysis. In addition to the software library, we present the results of three metrics that quantify balkanization: density, global degree centrality, and conditional probabilities of communication.

In Section 2 we present related research in the social sciences and computer science. In Section 3 we give an overview of the structure of Wikipedia. Section 4 discusses the dataset and map/reduce background. Section 5 explores the library implementation in detail before an analysis of the metrics used is presented in Section 6. A discussion of the results follows in Section 7, and Section 8 contains conclusions and possible further research.

2. Related Research

This thesis builds upon existing research on both balkanization and Wikipedia. There are three main categories of research: group dynamics, descriptive research, and metrics.

2.1 Group Dynamics

"Group dynamics" describes research about group formation and influences on groups. This research strongly motivated our work by giving examples which illustrated the real-world social processes that lead to balkanization.

McKenna et al. [20] examine the success and failure of newsgroup users at forming and reinforcing on-line relationships. They find that people like each other better if they meet on the Internet before meeting in person, highlighting the influence of the Internet on real-life relationships. Cheshire and Antin [7] review feedback mechanisms, like expressions of gratitude, that reinforce user participation on Internet sites. They look at "information pools," online collections of information, and analyze the effects of various feedback mechanisms on repeat contributions to these pools. They find that expressions of gratitude, reminders of past behavior, and ranking contributions influence readers towards further contributions.

Sunstein [22] discusses the effects of deliberation and social polarization on extremism. Groups make decisions based on in-group discussion and deliberation. Sunstein found that deliberation pushes groups toward more extreme, polarized viewpoints. Brewer and Pierce [5] investigated social identity complexity, the way in which individuals represent the relationships among their group memberships. The authors find that social groups which strongly overlap have members who are less tolerant and inclusive of outsiders.

The work of these authors makes it clear that balkanization is dangerous for overall social cohesion. In Wikipedia, deliberation over how to change an article is a kind of feedback mechanism. If these deliberative processes push users towards more extreme viewpoints, as

Sunstein suggests that they can, then Wikipedia could become balkanized. This represents the kind of situation that our research seeks to identify and prevent.

2.2 Descriptive Research of Wikipedia Behavior

We define descriptive research as research that analytically describes user behavior in Wikipedia. This type of research gave us background information on Wikipedia and revision history parsing. Ortega et al. [21] used Wikipedia's edit histories to determine statistics about Wikipedia such as the highest number of contributions per month, highest average revisions per month, and highest growth rate. Lewis et al. [17], performed a similar statistical analysis of Facebook [26]. While these authors parsed their data just to mine statistics, we are using our software library to parse out a persistent link graph. Although Ortega et al. also created a revision history parsing library, this work extends their methods by focusing explicitly on revision text. We use revision text for link mining and by tracking, across revisions, the exact ways in which users change pages. These methods allow us to find many more specific links in our data set.

2.3 Metrics

Many researchers have worked on developing balkanization metrics. Van Alstyne and Brynjolfsson [24] created a general model for user knowledge profiles of Internet users. This model could be applied to Wikipedia users, but it was not by these authors. These authors used a similarity metric to develop a measure for the overlap of user knowledge. This measurement does a good job of capturing knowledge balkanization amongst users.

White and Harary [27] used an advanced understanding of connectivity to look at conditional density in social networks. They look at how social cohesion can be discovered from looking at these two graph theory metrics. Although they do not, the metrics they use in their research could be applied to online social networks, in particular to Wikipedia.

Korfiatis and Naeve [16] analyzed article in- and out- degrees on Wikipedia. They used degree centrality, how central a vertex is to a graph, and article degree centralization, a measurement of the variability in individual degree centralities, to investigate article influence and credibility. If only a few articles have high influence, then the graph could be balkanized. On the other hand, if many articles have an equal amount of influence, the graph could be non-balkanized.

Capocci et al. [2] investigated the distribution of in- and out- degrees for Wikipedia articles by using a preferential attachment model. Preferential attachment is also known as "the rich get richer." They find that, although the preferential attachment model acts locally, user activity throughout Wikipedia can be described using it.

Jesus et al. [15] cluster a bi-partite graph of Wikipedia's users and articles on densely-connected graph subsections. They look for cliques of densely connected articles and users for clustering. Their bi-partite graph has two different, distinct sets for users and articles, with edges connecting them. However, there are no connections between just articles or between just users.

The primary goal of all these researchers is to analyze the balkanization of social networks. Instead of focusing on a single metric and investigating the results, we sought to construct a

platform from which a variety of metrical analyses could be performed. However, the three metrics we performed, density, global degree centrality, and conditional density, were all influenced by previous research. We used the density metric because we wanted to investigate a simple analysis that related to the metrics used by White and Harary in [27]. The degree centrality metric presented by Korfiatis and Naeve [16] directly influenced our use of global degree centrality. The conditional probabilities metric was inspired by discussions of user-specific balkanization that arose from the work done by Sunstein in [22].

When creating our link graph, we considered the designs used in the previous work reviewed above. Korfiatis and Naeve [16] used a directed, bi-partite graph of articles and users, Capocci et al. [2] used a directed graph of articles, and Jesus et al. [15] also used a bi-partite graph of articles and users. All of these graph models influenced the design of the link graph used in this research. The bipartite graph model was especially useful, although we did not use it, because the two sets have to be distinct, and we wanted to allow for edges between just users or just articles. We also decided not to use a directed graph, because articles and users should both be participants in the relationships between them. Furthermore, our research differs from that of these authors because of our coding of link information. By defining specific link types, we allow developers to easily analyze specific subsections of the graph. Additionally, by using the map/reduce parallel processing technique, this project allows developers without access to powerful systems to still perform analysis on Wikipedia's dataset. Instead of requiring a "big iron"-style computer, developers can run jobs on either a local Beowulf cluster or Amazon.com's ElasticMapReduce and S3 services. ElasticMapReduce in particular is something that almost all developers have access to.

3. Structure of Wikipedia

3.1 Overview of Wikipedia

Wikipedia has many unique features, but the two syntactic features that assist in graph parsing are *free links* and *namespaces*. *Free links* represent the classic HTML anchor tag, <a>. Other early wikis used CamelCase to denote an article link, but MediaWiki eschewed this "in favor of explicit link-markup with [[...]]" [28]. This unique syntax gives us an easy way to find links within a selection of Wikitext.

Namespaces define the scope and purpose of an article. Every article in Wikipedia belongs to one of the many namespaces. This project focuses on the default namespace, "Talk:", "User:" and "User talk:"

- default: Articles in the default namespace are informational encyclopedia articles.
- Talk: Articles in the "Talk:" namespace are known as "talk pages," and are used to discuss improvements to articles [12]. Talk: page names look like, "Talk:Istra" [14].
- User: User pages are unique articles that are maintained by a single user. They give the user a place to tell the community about themselves and allow the community to leave them comments. User: page names are prepended by "User:", such as in the page name, "User:Jesuislafete" [23].

- User talk: These pages are the talk pages for User: pages. In contrast to article talk pages, they are mainly used for public communication between users. Their names are formatted like, "User talk:Jesuislafete."

3.2 Wikipedia as a Graph

Wikipedia is a collection of users and articles that connect through various relationships. For example, a user may edit an article, an article may free link to another article, or a user may leave a note on another user's talk page. Mapping these relationships onto a graph allows us to analyze them using existing graph theory metrics.

The graph model has two main properties. First, the set of vertices contains two different vertex subsets, one for users and one for articles. Second, the set of edges is a commutative representation of the different kinds of relationships between users and articles.

The formal definition of the Wikipedia link graph G, as the example shown in Figure 1, follows:

$$G = (V, E) | V = (U, A)$$

where U is the set of user vertices, A is the set of article vertices, and $U \cap A = \phi$.

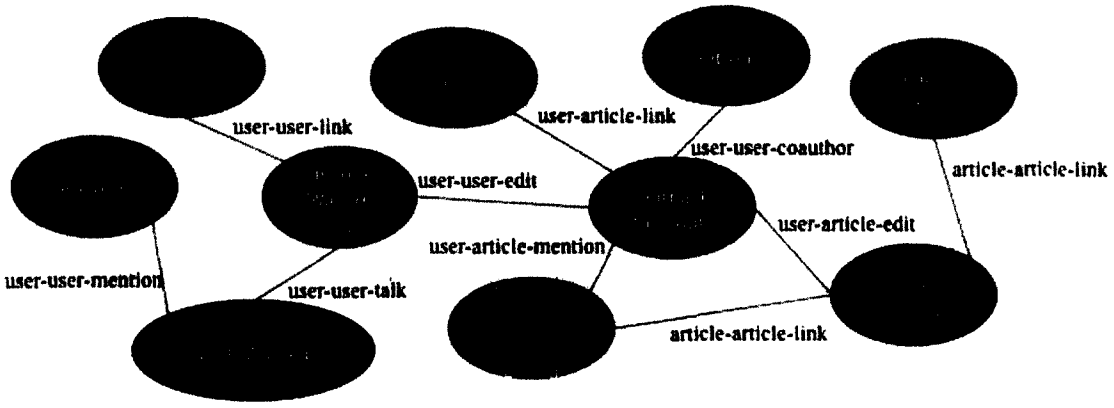

Figure 1 – Example Link Graph. A sample link graph between articles (blue) and users (red) with edges representing various relationships.

Figure 1 is an example link graph. ETST, HistoricWarrior007, Bogorm, 212.192.164.14, and Outback the koala are all Users; Canada, English language, United Kingdom, and Hargeisa Canadian Medical Center are articles. Each edge is a link between two vertices, but not every link represents the same type of relationship. The link descriptions next to each edge refer to the link types in Figure 2. Each row contains a link type, a description of how that link is formed, and an example from Figure 1. The link types represent different relationships between entities. These link types best represent the basic relationships between entities on Wikipedia.

Link Type	Link Description	Example from Figure 1
Article-Article (article A, article B)		
article-article-link	A free links to B	Article *English language* free links to Article *United Kingdom*
User-Article (article A, user U)		
user-article-edit	U edits A	User *Outback the koala* edits Article *United Kingdom*
user-article-link	U free links to A on its User: page	User *Outback the koala* free links to Article *Canada* on its User: page
user-article-mention	U mentions A on a Talk: or User talk: page	User *Outback the koala* mentions Article *Hargeisa Canadian Medical Center* on its User talk: page
User-User (user U, user V)		
user-user-edit	U edits V's User talk: page	User *HistoricWarrior007* edits User *Outback the koala*'s User talk: page
user-user-link	U free links to V on its User: page	User *HistoricWarrior007* free links to User *ETST* on its User: page
user-user-mention	U mentions V on a Talk: or User talk: page	User *212.192.164.14* mentions User *Bogorm* on User *HistoricWarrior007*'s User talk: page
user-user-coauthor	U coauthors with V on an article A	User *Outback the koala* coauthors with User *Yattum* on Article *United Kingdom*
user-user-talk	U talks with V on a Talk: or User talk: page	User *212.192.164.14* talks with User *HistoricWarrior007*

Figure 2 – Link Type Table. List of link types and examples.

4. Implementation

4.1 Dataset

Database dumps of Wikipedia are freely available to the public [29]. This research used a data dump from January 3rd, 2008. Each article is formatted using a custom XML schema and saved as an .xml file. Each .xml file is also compressed using 7zip, an open-source file archiver that uses the LZMA compression algorithm to achieve extremely high file-compression rates [1]. The size of the compressed data dump is 19GB.

An example of MediaWiki XML is shown in Figure 3. Each article's revision history is stored in a single file, inside of a <mediawiki> tag and a <page> tag. The article name and ID are stored in <title> and <id> tags. Following those is a series of <revision> tags in the order the edits were performed on the article. Each <revision> tag has four main elements and two optional elements. The four main elements are <id>, <timestamp>, <contributor>, and <text>, and contain the revision ID, the time and date of the edit, the author of the edit, and the textual content of the article as of that revision. The two optional elements are <comment> and <minor>, which contain any meta-comments left by the author and whether the edit is labeled as a minor edit (such as when a user reverts an article to an older revision).

```
<?xml version="1.0" encoding="utf-8"?>
<mediawiki xmlns="http://www.mediawiki.org/xml/export-
0.3/" version="0.3">
 <page>
  <title>Istra (disambiguation)</title>
  <id>2546600</id>
  <revision>
   <id>21902994</id>
   <timestamp>2005-08-26T19:30:08Z</timestamp>
   <contributor>
    <username>Mikkalai</username>
    <id>28438</id>
   </contributor>
   <text xml:space="preserve">'''Istra''' may refer to one of
the following.
*[[Istra]], a town in Russia.
*'''Istra''' is the Croatian and Slovenian name for [[Istria]].
{{disambig}}</text>
  </revision>
```

Figure 3 – Example MediaWiki XML. A single revision from the "Istra (Disambiguation)" page [14].

Not every link type can be parsed from a single article. For example, an article-article-link relationship could be found by looking at any page in the default namespace, as could article-user-edit connection, since both free link targets and contributor information are present in a single <revision> tag. However, the information needed to link the same article and user with an article-user-link connection (where a user's User: page free links to an article) can't

be found in this one article. Instead, parsing this link requires looking at a separate page in the User: namespace.

4.2 Map/Reduce

Map/reduce is a distributed computing method designed and implemented by Google, Inc. for use in their web crawling datacenters [10]. The original application was for Pagerank, a metric which measures the influence of individual webpages by looking at its in- and out-degrees [6]. Map/reduce was adapted for open-source use in Java applications by Apache and the Hadoop project [25]. For this project we used a Hadoop installation on St. Olaf College's MistRider cluster and Amazon.com's ElasticMapReduce service. MistRider is a 38-machine cluster made up of commodity machines; Elastic Map/Reduce is Amazon.com's flexible Hadoop cloud-computing platform.

The map/reduce paradigm is a parallel processing technique that utilizes two rounds of subprocesses to handle large amounts of data processing. The mapper tasks take key:value, tab-delimited text files as input. Each map task executes a data processing function defined by the developer and passes the results to the reducer tasks. Reducer tasks collate the information. Item organization, counting, and basic statistics are also common tasks for reducers.

Map/reduce is a natural fit for mining large data sets. The map/reduce process splits up input data equally among subprocesses, easing the load on each individual task. Additionally, map/reduce only requires developers to implement mapper and reducer functions, rather than an entire parallel processing structure. This allows developers to easily match up different mapper and reducer functions for different types of data processing. This flexibility works well for our software library. Map/reduce also supports the ability to feed the output of one job into the input of another, which enabled us to perform the iterative graph-parsing process outlined in Section 5.1 below.

The Hadoop map/reduce software library allows for easy deployment of jobs. It handles the distribution of input data, inter-process communication, and collating of output data for users. Hadoop employs a single JobTracker that administers the cluster and its own distributed filesystem, HDFS. Each job takes as input either a single file or directory in the HDFS, and outputs a directory of output files to the HDFS. All input, output, and communication between mappers and reducers is formatted as key-value pairs.

Each Hadoop job uses the JobConf class to handle job configuration. The JobConf class supports settings like the input directory, output directory, and output and input types. Each JobConf also requires the declaration of which mapper and reducer functions are being executed, as well as how many mappers and reducers to use.

Every Hadoop job has the same basic structure, as illustrated in Figure 4. First, the job is split into a number of map tasks and reduce tasks. Next, the input file(s) are pulled from the HDFS and distributed to the mappers. Each mapper runs its code and emits key-value pairs to the reducers. Reducers take mapper output and run their code, also emitting key-value pairs. Finally, the output data is written back to the HDFS.

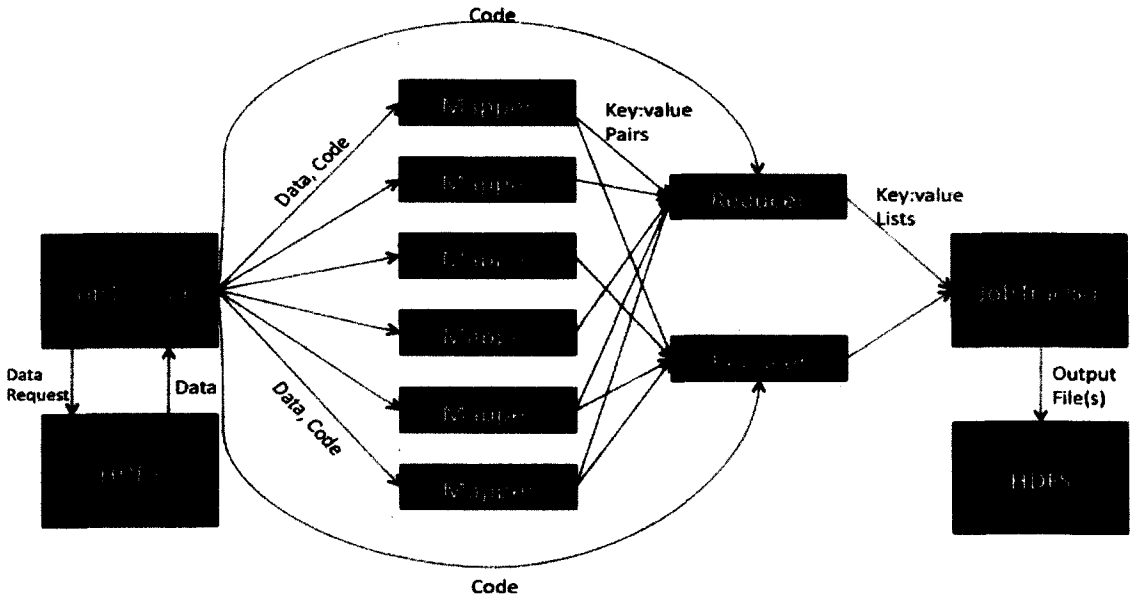

Figure 4 – Standard Hadoop Job Process. The normal Hadoop job lifecycle.

The map/reduce input files require a tab-delimited key-value text file, with one mapper's input on each line. A python script was used to convert the data dump filesystem into a single file, wikipedia.txt, consisting of filename:7zip-byte-stream key-value pairs. However, because some of the characters in the byte stream could be regular-expression escape sequences, such as '\r' and '\t', each of these also had to be double-escaped, so that '\r' became '\\r'. This ensured python would read '\r' as two different characters, '\' and 'r,' instead of as a carriage return.

5. Software Library

5.1 Overview

On Wikipedia, balkanization and social polarization can create divisions between authors, which in turn produce slow updates and articles that don't adhere to global editing norms. If we can detect balkanization, we can start working towards easing and, in the future, preventing it. We presented in Section 2.3 a number of researchers that have used graph theory metrics for analyzing balkanization on Wikipedia. This research seeks to expand on the work done by these researchers by implementing a library that creates a persistent link graph representation of Wikipedia by using its revision histories.

We created a software library to parse out the link graph into a coded, key:value, tab-delimited file. Since the link graph file persists beyond the use of the software library, researchers can perform analyses on it without having to parse it out each time. Furthermore, the file can be distributed on its own so that researchers without access to ElasticMapReduce or a computing cluster can still analyze the link graph.

Previous work, such as that done by Korfiatis and Naeve [14] and Ortega et al. [19] also created libraries for parsing Wikipedia revision histories. However, these libraries were built to fulfill specific goals. Ortega et al. [19] performed descriptive analysis that did not require a parsed link graph. Korfiatis and Naeve [14] did not parse a link graph that persisted beyond their metrical analysis.

Parsing the link graph is a five-stage process involving both map/reduce and non-map/reduce steps (Figure 5). Red boxes represent input and describe the input formats. Blue boxes represent individual functions. Each stage is summarized below.

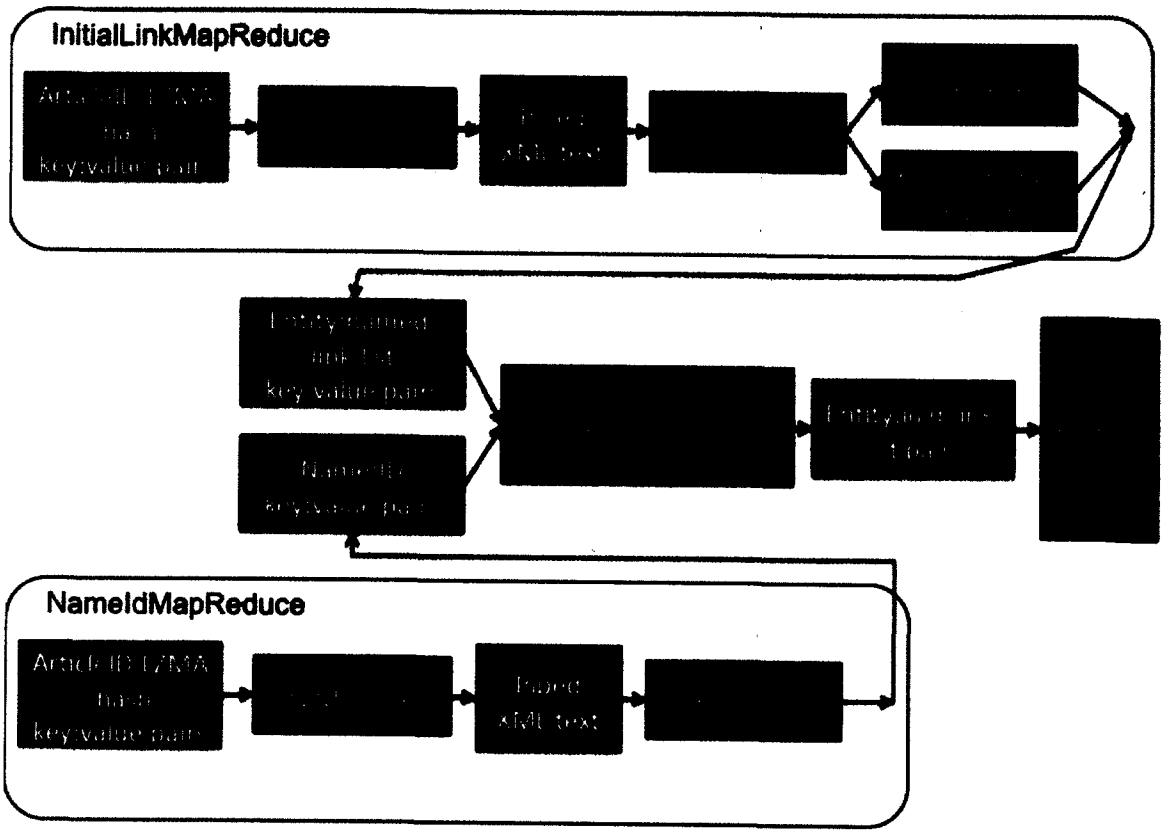

Figure 5 – Link Graph Parsing. The link graph parsing process, step-by-step. Red boxes are data formats, and blue boxes are WikiParser components.

Stage 1: *LzmaPipe* takes an input file containing the revision histories for a single article, decompresses it, and outputs it to a stream. *ArticleParser* converts the *LzmaPipe* output stream to a stream that handles the Revision XML format and uses it to parse out Article information and iterate through Revisions.

Stage 2: *LinkParser* processes the text of each Revision individually, looking for each type of Link in turn. *FingerprintingParser* also acts on the text of individual Revisions, but only

Talk: and User talk: pages. It finds neighboring-editor links for all the users communicating on the Talk and User talk: Articles.

Stage 3: *InitialLinkMapReduce* uses both the Stage 1 and Stage 2 processes to find the links present in the ID:xml key-value input file. *ArticleNameIdMapReduce* and *UserNameIdMapReduce* are contained in the *NameIdMapReduce* step in Figure 5. They both use just the Stage 1 processes to extract the IDs of all Entities in the same file.

Stage 4: *NameIdSubstitution* uses the stage 3 functions to create a copy of the link graph with only IDs instead of a mix of IDs and User and Article names. *CommutativeLinkMapReduce* isn't shown on Figure 5 in order to save space. It takes the link graph representation and applies all links commutatively.

Stage 5: This final link graph file can be used by metrics to evaluate Wikipedia. A description of the components required to perform this process follows.

5.2 Data Classes

The graph structure is represented by five main data classes: Entity, Article, User, Revision, and Link, as in Figure 6 below.

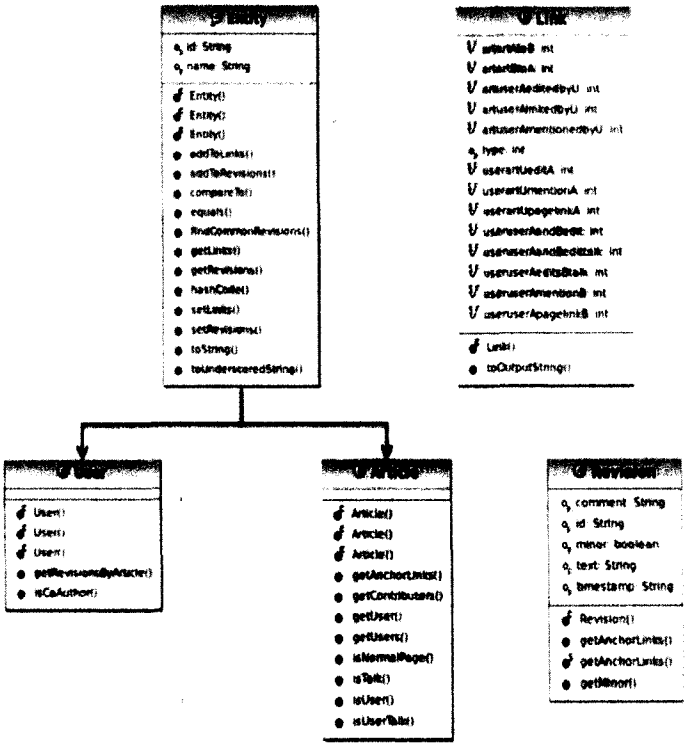

Figure 6 – Data Classes UML. UML diagram for WikiParser's data classes.

Entity is the parent class for Articles and Users, and contains a name, an Article or User id (which are unique by type), a list of Revisions, and a list of Links. The meaning of the Link list is the same for both Articles and Users, but the meaning of the Revision list is not. For Articles, the list of Revisions represents edits to that one article. For Users, the list of Revisions is the list of all edits performed by the user to any article. The Revision class represents the information contained in the <revision> XML tag: namely, a Revision id, a timestamp, the contributor, whether the revision is minor or not, any comments, and the entire text of the article as of that revision. The Link class represents the connection between two Entities, so it has two entity fields and a Link type.

5.3 LzmaPipe

An LzmaPipe takes as input an array of LZMA-compressed bytes and outputs a decompressed PipedInputStream. It uses an additional thread to provide a persistent stream object for as long as its contents are needed. This allows for the decompression of very large files without requiring large amounts of memory. This is used for the largest files which are, when decompressed, too large to reliably fit in memory. For example, "Wikipedia:Administrator's noticeboard/Incidents" is 37 MB compressed and 51 GB decompressed, and "Wikipedia:Reference desk/Miscellaneous" is 18 MB compressed and 19 GB decompressed.

5.4 ArticleParser

The ArticleParser class, Figure 7, parses Article information from an input stream. ArticleParser provides methods for reading information from an InputStream containing a MediaWiki .xml file. Each map task creates the InputStream using an LzmaPipe instance. The ArticleParser then converts the input stream into an XMLStreamReader, which enables it to traverse the file text through individual tags.

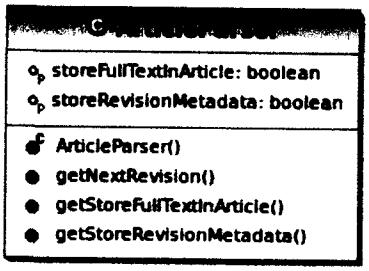

Figure 7 – ArticleParser UML. UML diagram for the ArticleParser class.

There are two primary methods in ArticleParser: *getArticle()* and *getNextRevision()*.

Article getArticle(): This method instantiates an Article object inside of the ArticleParser and returns it. Each call to *getNextRevision()* adds a new Revision object to the local Article instance. An Article object with all of its Revisions can be retrieved by calling *getNextRevision()* until it returns null.

Revision getNextRevision(): This method returns the next Revision in the XMLInputStream, or null if there are none left. *getNextRevision()* uses the private *matchElement()* method to find each of the six Revision fields, "id," "timestamp," "contributor," "minor," "comment,"

and "text." Each field is stored as a String, except for "minor," which is saved as a boolean, and "contributor," which is stored in a User object that is returned by the private method *readContributor()*.

This class was originally designed to hold the entire Article object, with complete Revision and User lists, in memory. However, the individual MistRider cluster machines did not have enough memory to handle the largest decompressed .xml files. Instead, each mapper uses an LzmaPipe to create an InputStream for the ArticleParser.

A related problem also arose: even though the entire revision history was no longer being held in memory, large files with lots of revisions could still use up all the memory available to a single map task. MistRider mappers do not have very much memory available, but nodes on Amazon's S3 can be configured to have more than enough. There are two options that can be set in ArticleParser to handle this variation in memory availability. One option is to determine whether the text of every revision is stored in the local Article instance. This can help keep down memory overhead. A second option sets whether any Revision objects are stored in the local Article instance. If this is set, the local Article instance never has any Revisions added to its Revision list. Although this has the effect of causing *getArticle()* to only retrieve the Article's ID and name, it also nearly eliminates memory overhead issues.

5.5 LinkParser

The LinkParser class, Figure 8, finds the Links emanating from a particular Entity. LinkParser finds links by looking at an Article created by ArticleParser. If the article is in the User: or User talk: namespaces, then the Entity is a User. If the article is in the default or Talk: namespaces, then the Entity is an Article object. The *findLinks()* method uses LinkGenerators to find and return a list of Links for each Revision.

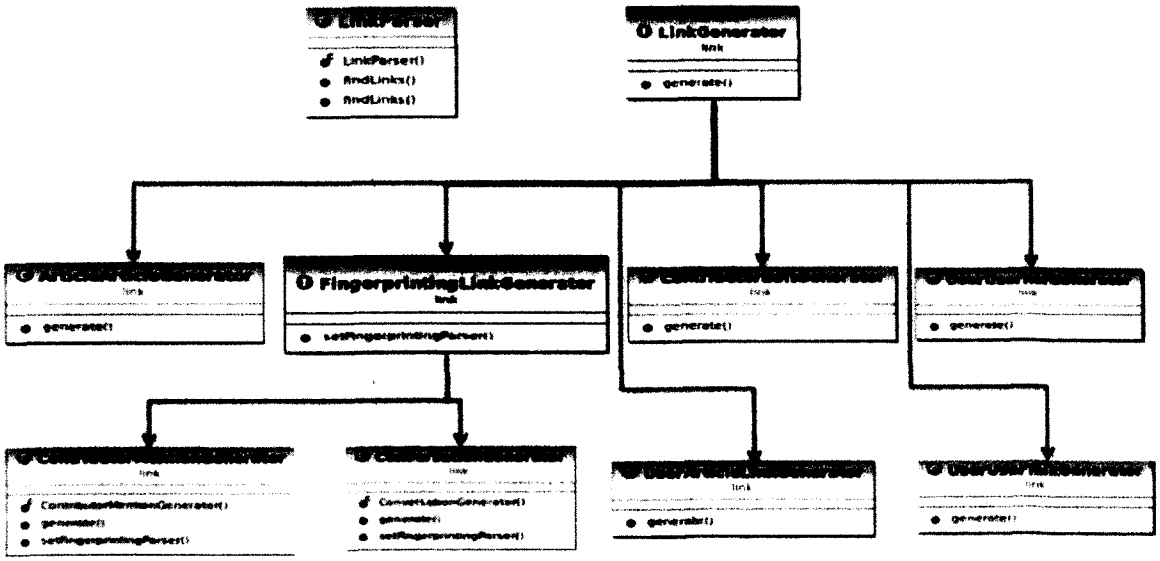

Figure 8 – LinkParser UML. UML diagram for the LinkParser class and the LinkGenerator family of classes.

LinkGenerators are classes that implement the LinkGenerator interface. Each one forms a list of Links meeting specific criteria. The LinkGenerators used for parsing the link graph have rules taken from the list of Link types reviewed earlier.

Due to Java 6's elegant for-each syntax, LinkGenerators provided a very simple way to look for each individual Link type. There have been issues with not every Link being found, or the Link being of the wrong Link type, but by compartmentalizing the functions for finding Links into the LinkGenerator classes, logic bugs were much easier to find and resolve.

5.6 FingerprintingParser

FingerprintingParser, Figure 9, finds user-user-talk links. In contrast to coauthoring an Article, Users communicate with each other when they talk on a Talk: or User talk: page. The FingerprintingParser discovers which user pairs have had a conversation by finding each User's unique contributions to a Talk: or User talk: page. It then finds the contributions nearby it, which indicate other authors the User has talked to.

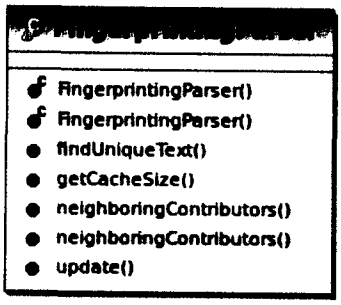

Figure 9 – FingerprintingParser UML. UML diagram for the FingerprintingParser class.

The FingerprintingParser implements a rolling hash to find unique text. A rolling hash works by setting up a "window" of text that the parser is currently looking at. All the words and delimiters inside the window are sent to the *hash()* method, which returns an almost-unique integer fingerprint. An internal hashmap stores all the fingerprints as fingerprint-Revision pairs. The rolling hash window iteratively looks at the entire document, yielding a number of fingerprint-Revision pairs.

The rolling hash window finds new words when it passes over them. However, if the words start out in the window, it can't determine if they are new, unique text or old. Changes in words at the beginning of the text are detectable, because the window will produce an altered fingerprint, but we can't determine which specific words changed. In order to pick up these changes, FingerprintingParser prepends and concatenates a special marker onto the text of every Revision that it analyzes. By putting a number of markers equal to the size of our rolling hash window on the beginning and end of every Revision text, the window can hash words at the beginning of the text and determine whether they are new or not. A small example of the rolling hash function, with special markers, is shown in Figure 10.

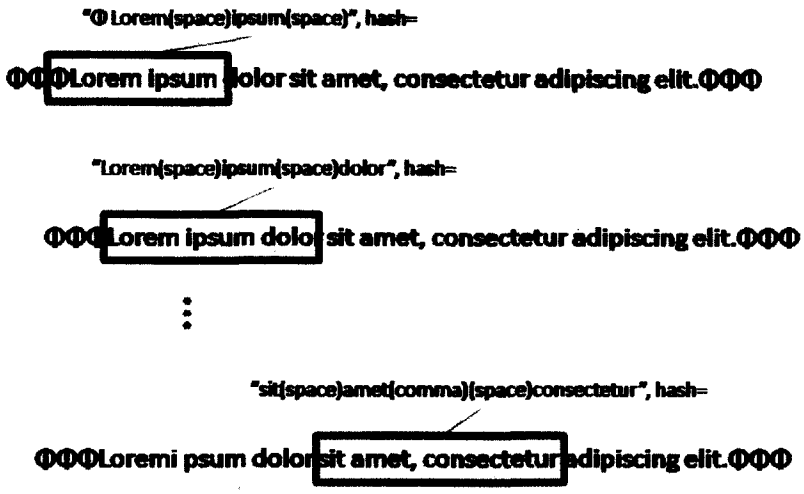

Figure 10 – Windowing. An example of the rolling hash's "windowing" functionality. The first window shows the beginning of a section of text including markers, the second window shows just the beginning of the text, and the third showcases the middle of the text: note that the comma and space in the third window count as one 'word' to the function.

The hashmap that stores the fingerprint-Revision pairs acts as a cache for the fingerprints. When the maximum number of key-value pairs is reached, the hashmap automatically removes the oldest entry. This functionality enables us to configure how much memory we are using for the FingerprintingParser.

ArrayList<String> findUniqueText(Revision revision): This method finds the unique text that a single Revision adds to the Article as a whole. The *findUniqueText()* method does not add any newly-discovered fingerprints to the hashmap. Instead, it uses the existing contents of the hashmap to determine whether a given fingerprint is new. This method also doesn't check for fingerprint ownership, so it should only be called on a Revision before that Revision is passed to the *update()* method, which does add fingerprints to the hashmap. If this is not done in the correct order, *findUniqueText()* may not find actual unique text for a given Revision because that Revision's fingerprints are already in the hashmap.

findUniqueText() travels iteratively through the document to find fingerprints. It also uses a state transition to determine when a new word has been found. A window can be looking at either old text or new, unique text. The state transitions are as follows.

- Old text → new text: If the window moves from looking at old text to looking at new text, then a new word is necessarily at the end of the window, and the window is moved forward to the first word of new text, which is also added to a String of unique text.
- New text → new text: If the state of the window moves from new text to new text, then the new first word of the window is also added to the unique text String.

- New text → old text: If the window stops observing new text and comes across fingerprints that are already stored in the HashCodeMap, then that section of new text is finished, and can be added to the ArrayList<String> that this method returns. This is equivalent to the window moving from new text to old text.
- Old text → old text: If the window continues to observe old text, then nothing happens.

The ArrayList of new text sections compiled during the iterative process is returned after all the text has been fingerprinted.

ArrayList<User> neighboringContributors(Revision rev, ArrayList<String> uniqueText): The *neighboringContributors()* method uses what is called a "neighborhood" to determine what users have communicated with a particular Revision's author. A "neighborhood" is a set amount of space located before, or after, text that a particular Revision added. All users who edited text within a piece of text's neighborhood are "neighbors" for the User who wrote the text. If User B's Revision is being considered, and User A has text within the neighborhood of text that User B has written, then User A is added to the ArrayList<User> for User B.

This method works iteratively, although not in the same manner as *findUniqueText()*, because this method has both the full Revision text and the small strings of unique text returned by *findUniqueText()*. For each section of unique text, the neighborhood before and after that text are both inspected. The rolling hash window goes through each of these neighborhoods. When the author of those text sections differs from the author of the original Revision, the contributors of the text are added to the ArrayList<User>.

This method can only return neighboring contributor information to the extent that the edit history of the Article is known to the FingerprintingParser's hashmap. Consider the situation, shown in Figure 11, where Revision 1 adds a few lines of text to an Article, Revision 2 adds a few lines at the end of Revision 1, and Revision 3 adds 12 words in the middle of Revision 1's text. If this method were passed Revision 2's unique text before passing Revision 3 to *update()*, then it will only return the author of Revision 1. However, if this method were called on Revision 2 after Revision 3's fingerprints are added to the hashmap, then the method will return the authors of both Revision 1 and Revision 3. Furthermore, if a fourth Revision just removes all of Revision 1's text, then passing Revision 2 to this method yields only the author of Revision 3.

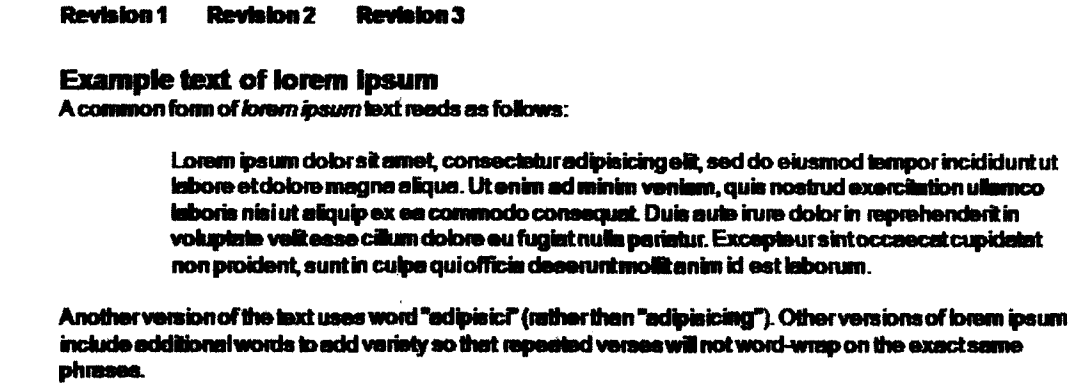

Figure 11 – Sample Text by Revision. A selection of sample text from the article on lorem ipsum, split up along arbitrary revision lines [18].

This ambiguity is useful because we can use this method to find two different sets of users. We can find users who are actively talking as of a particular Revision, or find all the users who have ever talked in an Article, regardless of whether their conversation has been saved.

5.7 InitialLinkMapReduce

InitialLinkMapReduce is the first map/reduce step in parsing the link graph. This mapper function consists of 3 steps: unescaping the input line, decompressing it through an LzmaPipe, and finding the links. The series of modules just reviewed represent the entirety of the InitialLinkMapReduce step from Figure 5.

Due to the double-escaping performed in the data formatting step, the value of each line needs to be unescaped back to its original state before it can be decompressed without errors. After it is unescaped, the compressed text is passed into an LzmaPipe, which creates an input stream in a separate thread that can be accessed iteratively. Each article needs to be streamed because the LZMA compression algorithm is very effective, and filesizes balloon when they are decompressed. For example, the article "Wikipedia:Administrator's noticeboard/Incidents" has a compressed filesize of 37MB, but a decompressed filesize of 51GB, which is far too large to be able to fit in the memory available to a single mapper.

Once the pipe is created, the ArticleParser class takes the LzmaPipe's output stream and reads in the article a revision at a time. This allows each individual revision to be processed on its own. The LinkParser class is then used to find all the Links between the article and other entities. If the article is a User: or User talk: page, the Links are created between the article's owner and other entities instead. Each Link is emitted to the reducer, which aggregates the Links into article-or-user-ID:list-of-links pairs, as in Figure 12.

a2546600 a01Istria a01Istra_Resivoir a01Istra_River u06Mikkalai u06Markussep u06Jesuislafete u06Goudzovski\n

Figure 12 – Sample Link Output. An example of the InitialLinkMapReduce output format.

Every Link is outputted in the following format: {a,u}[link type][name]. Each link type has been coded into a numerical value to make them easier to compare. The first letter denotes the type of the targeted Entity, the next two digits refer to the type of the Link, and then the name of the article follows. The letter at the beginning of the line denotes the type of the parsed entity. Here, article 2546600, "Istra (disambiguation)," has links to a variety of Entities, including an article-user-edit Link to "Markussep" and an article-article-link Link to "Istra-Resivoir."

5.8 NameIdSubstitution

Note in Figure 12 that only "Istra (disambiguation)" is referred to by an ID. This is because we can't get IDs for the other Entities just by looking at this one article. The next step to parsing the link graph is to substitute IDs for the article names and user names emitted by the InitialLinkMapReduce. Using IDs universally allows us to apply links commutatively in the next step. This step builds on the InitialLinkMapReduce and NameIdMapReduce steps in Figure 5.

ArticleNameIdMapReduce and UserNameIdMapReduce output name:ID pairs for Articles and Users, respectively. Each ID value is encoded with the type of the entity at the beginning of the ID. After this job runs, NameIdSubstitution, a non-map/reduce program, is run to substitute in each ID. NameIdSubstitution creates one HashMap for each type of Entity, and uses the output encoding from ArticleNameIdMapReduce and UserNameIdMapReduce to determine which ID is an Article ID and which is a User ID. Each line in the link graph file is rewritten to a new file by looking up each link's name in the appropriate HashMap and substituting the correct ID.

5.9 CommutativeLinkMapReduce

The final step in creating the link graph is the CommutativeLinkMapReduce job. This step is an additional step not shown on Figure 5. It reads in each line of the InitialLinkMapReduce and outputs links from each former link target to the key entity as well as the original links. This is then aggregated by the reducer and output, creating the final representation of the link graph.

6. Metrics

The metrics reviewed below each produce different evaluations of the connectedness of the link graph. These evaluations each tell us something about the presence of balkanization in Wikipedia. Each metric takes the parsed link graph as input and outputs real-valued results.

6.1 Global Density

Global density is the number of edges in a graph divided by the number of possible edges in the graph. It measures how connected a graph is by comparing it to an ideal, completely connected graph. A densely connected graph shows little signs of balkanization, because users and articles are widely connected, and not constrained to small groups. A not very connected graph would be balkanized, as would a graph with pockets of connectedness that aren't themselves interconnected. Both of these graphs would have low global density. However, a graph with pockets of connectedness that are themselves relatively connected would have high global density, and would not be balkanized. Further analysis of the link graph would need to be performed in order to determine what the underlying structure of the graph is. The formula for density is as follows:

$$D(G) = \frac{\|E\|}{\|V\| * (\|V\| - 1)} \quad (1)$$

A density close to 0 means that the graph has very few edges and is not very well-connected. A density closer to 1 represents a graph that is very well connected and has close to the full set of possible edges. A balkanized graph would have a global density score close to 0, since it would contain very few of the total possible links.

We decided to use the density metric because we wanted to perform a simple analysis related to the metrics outlined for general application by White and Harary in [27]. They gave general formulae for a measure of group connectivity based on how many vertices needed to be removed for a group to become disconnected. They then expanded that measure to approximate conditional density, which proportionally rates groups based on how many edges they would need to increase their connectivity rating. We decided that an analysis of a simple density function would serve as a proof-of-concept for more complicated models, such as conditional density. However, the data needed to perform any one of these analyses is present in the link graph. Our implementation for calculating *D(G)* reads in the link graph file one line at a time and keeps track of $\|E\|$ and $\|V\|$, which we then use to calculate density.

6.2 Global Degree Centrality

Global degree centrality measures how much a graph resembles a star-shaped network, with one central vertex and many fringe vertices connected only to the center. On a global scale, this measures balkanization by describing the overall structure of the link graph. If the graph is generally star-shaped, with a small number of central vertices, it has a chance of being balkanized. If, on the other hand, the graph is not very star-shaped, that means the links could be evenly distributed throughout the network, suggesting a lack of balkanization. However, this could also mean there are many central vertices, each with its own star-shaped or evenly-distributed graph subsection. To determine the precise structure, the global degree centrality metric would need to be applied to smaller subsections of the graph. We did not take this step for this research due to time constraints.

The degree centrality of a single vertex is the number of edges it has divided by the number of edges it could have; a degree centrality close to 1 represents a vertex that is well-connected to all the other vertices, and a degree centrality of 0 denotes a vertex that is not

well connected. Cd(G) represents the global degree centrality, where V* is the vertex with the highest individual degree centrality, as calculated in (2).

$$Cd(V_i) = \frac{\deg(V_i)}{\|V\| - 1} \tag{2}$$

$$Cd(G) = \frac{\sum_{i=1}^{|V|} Cd(V^*) - Cd(V_i)}{\|V\| - 2} \tag{3}$$

The numerator of equation 3 is the degree centrality of each vertex compared to V*, with these differences then summed, and the denominator is the number of edges outside of the pair being compared. This formula also yields a result between 0 and 1, with a result close to 1 meaning the graph has a shape with a few central nodes and many nodes with few connections (see Figure 13), and a result close to 0 representing a universally well-connected graph, as in Figure 14.

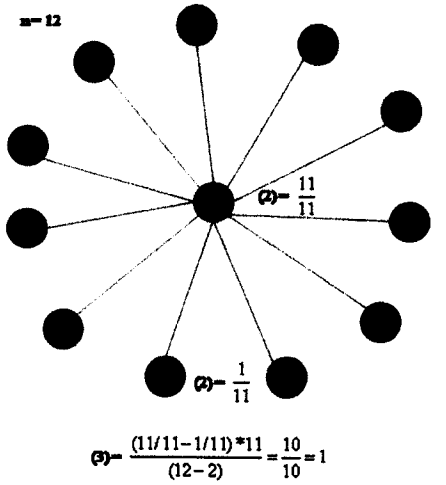

Figure 13 – Star-shaped Graph. An example of a star-shaped network for the global degree centrality metric (equation 3).

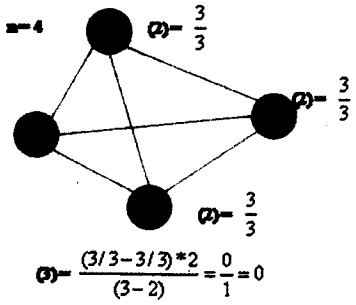

Figure 14 – Fully-connected Graph An example of a fully-connected network for the global degree centrality metric (equation 3).

The social significance of Figure 13 and Figure 14 should be quite plain. If the vertices in Figure 13 were users, then outside vertices could only communicate with each other through the central node. This could lead to miscommunication between users. In Figure 14, each vertex would be able to communicate with every other vertex. The social space represented by Figure 13 is much more balkanized than the space represented by Figure 14. However, a more realistic balkanized graph would be made up of many Figure 14-like star-shaped graphs.

The degree centrality metric presented in Korfiatis and Naeve [16], which analyzed in- and out- degrees of articles on Wikipedia, influenced our choice of global degree centrality.

They used the metric for creating a ranking mechanism, rather than to test for balkanization. As a metric that focuses on individual nodes, as opposed to edges, global degree centrality also balanced well with the density metric reviewed above, which was another reason we chose to use it for this research.

The implementation of degree centrality reads in the link graph file line by line, creating an ID:degree HashMap of vertex degrees. These stored values are then used to calculate Cd(G).

6.3 Conditional Probability

The conditional probability metric evaluates the chance that a user talks with a user they've coauthored with versus with a user they have not coauthored with. This metric focuses on the vertices which represent users and the edges that represent links between users.

These conditional probabilities were directly inspired by discussions about user-specific balkanization and communication. Specifically, we wanted to determine whether communication was affected by coauthorship, so we developed these conditional probabilities to evaluate those ideas.

Users can connect directly with each other in one of two ways: coauthoring articles together and discussing on Talk: and User talk: pages. In the article-editing paradigm, these two actions are related. If a user wants to make improvements to an article, they are likely to discuss those changes with other users working on the article beforehand. If a user is discussing changes with other users, it's likely that they'll coauthor an article with those users.

If a pair of users has a high probability of talking to each other given that they have coauthored versus given that they have not coauthored, then we know that coauthoring improves the chance that users talk.

This metric uses Bayes' Theorem of conditional probability to look at the probabilities involved in editing articles with and talking to other users, as in equation 4.

$$P(talking \mid coauthoring) = \frac{P(coauthoring \wedge talking)}{P(coauthoring)} \qquad (4)$$

P(talking | coauthoring) is calculated from the probabilities for a pair of users to coauthor, talk, and coauthor and talk, as shown below.

$$P(coauthoring) = \frac{\#ofPairsWhoCoauthor}{\#ofTotalPairs} \qquad (5)$$

$$P(talking) = \frac{\#ofPairsWhoTalk}{\#ofTotalPairs} \qquad (6)$$

$$P(coauthoring \wedge talking) = \frac{\#ofPairsWhoCoauthorAndTalk}{\#ofTotalPairs} \qquad (7)$$

The count values needed for equations 5, 6, and 7 are culled from the final link graph output. The number of coauthor pairs can be found by pairing up users and counting the number of user-article-edit Links across every article with more than one contributor. The number of

user pairs who talk can be found by looking for user-user-talk links. For this metric, we sampled down to one hundredth of the userbase, because the memory overhead for remembering every user pair was too large.

7. Preliminary Results

7.1 Library Implementation

The library successfully parsed the entire Wikipedia revision history. Each step in parsing the link graph took between an hour and an hour-and-a-half to complete on both ElasticMapReduce and MistRider, which operate with 26 and 38 CPUs, respectively. Each run produced output with sizes in excess of 4 GB; storing this amount of data is not an issue for modern computers. In addition, S3 and ElasticMapReduce are publicly available, making this an easily usable platform from which to run balkanization metrics.

7.2 Density

We found that there were 436,489,820 edges and 39,214,905 vertices in the link graph, yielding a global density (1) of approximately 2.8384*10^-7. That means the graph is very sparse, with few links compared to the total number of possible links.

The link graph is composed of both articles and users, and the set of edges represents links between articles, links between users, and links between articles and users. It is certainly possible that: 1) density by link type is not uniform, and one or more link types are negatively affecting the global density; or 2) that individual subsections of the full graph, such as just user vertices, are far less dense than others, which would also affect the global density value. It is our intuition that this first hypothesis is true, and we can see in Figure 15 evidence that link type affects density. Links of type user-user-link and user-user-mention are very sparse compared to links of type article-article-link.

Link Type	Edge Counts	Density
article-article-link	275572609	1.7920*10^-7
user-article-edit	44207941	2.8747*10^-8
user-article-link	76705564	4.9880*10^-8
user-article-mention	24988551	1.6249*10^-8
user-user-edit	0*	0.0*
user-user-link	2125554	1.3822*10^-9
user-user-mention	10132905	6.5892*10^-9
user-user-coauthor	0*	0.0*
user-user-talk	1	6.5028*10^-16

Figure 15 –Density Values by Link Type. A table of edge counts and density values, by link type. *User-user-edit and user-user-coauthor were not calculated.

Due to an error in the parsing process, the density values for user-user-edit Links and user-user-coauthor Links were not calculated. The addition of these links would affect global density. However, given the scale of the other link type densities, it is our intuition that global density would not be dramatically altered.

Regardless, further research is needed to prove or disprove either of these hypotheses. Overall, this metric does tell us something about balkanization: when taken as a whole, Wikipedia has a low amount of global connectivity, which suggests balkanization into smaller subgroups.

We also do not know the scale of these values in relation to other real-world examples. For example, taking subsections of the link graph by topic area may yield differently scaled results. This would tell us more about the global density value as well.

7.3 Degree Centrality

We found a global degree centrality (2) score of 0.02504. Average degree per vertex was 13.17 and average degree centrality for each vertex was 3.3578*10^-7. The top ten degree and degree centrality scores are shown in Figure 16.

Ranking	ArticleName	Degree	Degree Centrality
1	Wikipedia:Sandbox	982100	0.02504
2	Wikipedia:Version 1.0 Editorial Team/Assessment log	865523	0.02207
3	Cyde/Archive	638215	0.01627
4	Help:Reverting	610332	0.01556
5	User 433328	591237	0.01508
6	Wikipedia:Introduction	590275	0.01505
7	Wikipedia:Vandalism	559524	0.01427
8	User 1215485	552085	0.01408
9	Wikipedia:Welcome, newcomers	551418	0.01406
10	Wikipedia:Version 1.0 Editorial Team/Biography articles by quality log	452387	0.01154

Figure 16 –Top Ten Degree Centralities. A table of the top ten degree counts and degree centrality values, by vertex.

As a measure of how much the graph deviates from a basic star shape, this value is telling us that the link graph is not star-shaped. On the surface, this appears to be evidence of a lack of balkanization. However, the top-ten results illustrate another important point: the range of possible values is only [0, 0.025]. Global degree centrality compares the individual degree centrality of vertex pairs. If you recall, in (2), V^* represents the most well-connected vertex, and it is compared with V_i, every other vertex. If the most well-connected vertex doesn't have a high individual degree centrality, then the range of other possible individual-degree-centrality values is small. The largest degree-centrality is only ~ 0.025, and the mean indicates that most vertices have $Cd(V_i)$ close to 0.

This result does not preclude the existence of balkanization in the graph. Rather, the results simply don't say very much about balkanization on a global scale. Once again, we have no sense of the scale of these values, so we don't know what these results could be telling us. Further research would need to be done on smaller portions of the link graph in order to determine if degree centrality varies throughout the network, and if this metric can be used to detect balkanization.

7.4 Coauthor/Conversation Probabilities

After sampling down to one hundredth of the data set, our sample link graph contained 110,257 total users, 12,156,495,792 possible user pairs, 2,757,654 users pairs who coauthored together, and 1,366 user pairs who conversed on Talk: or User talk: pages with one another. The probability that a pair of users coauthored an article together was 2.2685*10^-4 (equation 6), the probability that a pair of users talked with one another on a Talk: or User talk: page was 1.1237*10^-7 (equation 7), and the probability that a pair of users did both was only 1.9743*10^-9 (equation 8). The probability that a user pair talked given that they coauthored an article together was 8.7030*10^-6 (equation 5), and the probability that a user pair coauthored an article given that they had talked with each other was 0.01757 (equation 4).

Both conditional probabilities were higher than their unconditional counterparts, which points towards communication between users yielding positive results for the community as a whole. Specifically, users who coauthored were also almost 11 times more likely to talk given that they had coauthored, as compared to all users who talked.

These results tell us two things: firstly, the chance that users communicate is fairly slim; secondly, communication and co-participation seem to be correlated. This second conclusion has multiple implications: it can be taken as evidence against balkanization, because users who interact are much more likely to participate in the site, or, alternatively, it can be taken as evidence of balkanization, because those users still aren't interacting with users outside of their editing group. Research looking at these probabilities by topic area could yield results that shed further light on this problem.

8. Conclusions and Further Research

This paper has detailed the basis for and implementation of a library that uses the Hadoop map/reduce system to create a graph of Wikipedia by parsing Wikipedia revision histories. Furthermore, metrics have been presented which parse the link graph and evaluate it. These contributions allow future researchers to easily investigate and evaluate the presence of balkanization on Wikipedia.

We found evidence of balkanization in the results for the global density metric. The global degree centrality and conditional probability metrics did not find explicit evidence against balkanization. The presence of one metric finding evidence of balkanization suggests that further research should be undertaken. For density and global degree centrality, we found that the scale of the results is unclear. It is important that further research using these balkanization metrics uses a variety of subsections of the English language link graph in order to clear up the real-world scale of these results.

Other non balkanization-metric-based projects can also be completed using this library. For example: visualizing the graph; calculating other graph theory metrics, such as structural cohesion or the clustering coefficient; exploring probabilistic predictive models; and determining interesting statistics. With a Hadoop installation and sufficient processing time, many applications can be created which use this library. By taking advantage of services like

Amazon's Elastic Map/Reduce and S3, even those without access to a dedicated cluster can use the methods and tools presented here to analyze Wikipedia.

Finding balkanization on Wikipedia is important, not only to ensure the quality of Wikipedia articles, but also to fine-tune metrics and techniques that can be used to detect balkanization and social polarization in other social and informational networks.

9. Acknowledgements

I would like to thank Professor Shilad Sen of Macalester College first and foremost, for his immense help in envisioning this project and its implementation. I would also like to thank Professor Libby Shoop of Macalester College for her help with Hadoop and editing, as well as Professor Dick Brown of St. Olaf College for the invaluable freedom to use the St. Olaf College MistRider beowulf cluster. I would also like to thank Professor Andrew Beveridge of Macalester College for his encouragement and time, and Neil Hilborn of Macalester College for his help with editing.

10. Bibliography

1. "7-zip – Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/7-zip. Accessed: Dec. 20, 2009. Updated: Dec. 18, 2009.

2. Capocci et al. *Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia.* Physical Review, Vol. 74, No. 3, 2006.

3. "Balkanization." http://en.wikipedia.org/wiki/Balkanization. Accessed: Apr. 4, 2010. Updated: Feb. 17, 2010.

4. "Bart Stupak Decides to Retire After Vitriolic Healthcare Debate." http://chattahbox.com/us/2010/04/09/bart-stupak-decides-to-retire-after-vitriolic-healthcare-debate/. Accessed: Apr. 15, 2010. Updated: Apr. 9, 2010.

5. Brewer, Marilynn B. and Kathleen P. Pierce. *Social Identity Complexity and Outgroup Tolerance.* Urbana: Sage Publications, 2009.

6. Brin, Sergey and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." http://infolab.stanford.edu/~backrub/google.html. Accessed: Dec. 20, 2009. Updated: Feb. 4, 2009.

7. Cheshire, Coye and Judd Antin. *The Social Psychological Effects of Feedback on the Production of Internet Information Pools.* Bloomington: Journal of Computer-Mediated Communication, Vol. 13, 2008. 705-727.

8. "Conservapedia." http://www.conservapedia.com/Main_Page. Accessed: Apr. 4, 2010. Updated: Apr. 8, 2010.

9. "Courthouse News Service." http://www.courthousenews.com/2010/03/23/25826.htm. Accessed: Apr. 15, 2010. Updated: Mar. 23, 2010.

10. Dean, Jeffery and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." Google, Inc.: 2004.

11. "Health care reform debate in the United States." http://en.wikipedia.org/wiki/Health_care_reform_debate_in_the_United_States.

12. "Help:Talk page – Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Wikipedia:Talk_page. Accessed: Dec. 20, 2009. Updated: Dec. 20, 2009.

13. Himelboim, Itai. *Reply distribution in online discussions: A comparative network analysis of political and health newsgroups.* Bloomington: Journal of Computer-Mediated Communication, Vol. 14, 2008. 156-177.

14. "Istra (disambiguation)." http://en.wikipedia.org/wiki/Istra_%28disambiguation%29. Accessed: Apr. 15, 2010. Accessed: Apr. 15, 2010. Updated: Feb. 12, 2010.

15. Jesus, Rut et al. *Bipartite Networks of Wikipedia's Articles and Authors: a Meso-level Approach.* Orlando: ACM, 2009.

16. Korfiatis, N. and A. Naeve. *Evaluating wiki contributions using social networks: A case study on Wikipedia.* Stockholm: Growth.

17. Lewis, Kevin et al. *Tastes, ties and time: A new social network dataset using Facebook.com.* Cambridge: Elsevier B.V., 2008.

18. "Lorem ipsum." http://en.wikipedia.org/wiki/Lorem_ipsum. Accessed: Apr. 15, 2010. Updated: Apr. 8, 2010.

19. "Main page – dKosopedia." http://www.dkosopedia.com/wiki/Main_Page. Accessed: Apr. 4, 2010. Updated: Mar. 2, 2010.

20. McKenna, Katelyn Y. A. et al. *Relationship Formation on the Internet: What's the Big Attraction.* New York: Journal of Social Issues, Vol. 58, No. 1, 2002. 9-31.

21. Ortega, Felipe and Jesus M. Gonzalez-Barahona. *Quantitative Analysis of the Wikipedia Community of Users.* Montreal: ACM, 2007.

22. Sunstein, Cass R. *The Law of Group Polarization.* Chicago: University of Chicago Law School, 1999.

23. "User: Jesuislafete – Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/User:Jesuislafete. Accessed: Dec. 20, 2009. Updated: May 29, 2008.

24. Van Alstyne, M. W. and E. Brynjolfsson. *Global Village or CyberBalkans: Modeling and Measuring the Integration of Electronic Communities.* Cambridge: MIT, 1997.

25. "Welcome to Apache Hadoop!" http://hadoop.apache.org/. Accessed: Dec. 21, 2009. Updated: Jul. 16, 2009.

26. "Welcome to Facebook." http://www.facebook.com/. Accessed: Apr. 15, 2010. Updated: Apr. 15, 2010.

27. White, Douglas R. and Frank Harary. *The Cohesiveness of Blocks in Social Networks: Node Connectivity and Conditional Density.* Washington, DC: Sociological Methodology, Vol. 31, 2001. 305-359.

28. "Wikimarkup – Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Wikimarkup. Accessed: Dec. 20, 2009. Updated: Dec. 19, 2009.

29. "Wikimedia Downloads." http://download.wikimedia.org/. Accessed: Apr. 15, 2010. Updated: Mar. 1, 2010.

30. "Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Main_Page. Accessed: Apr. 4, 2010. Updated: Apr. 7, 2010.