The Internet of Edits: *Understanding the Wikipedia edit network*

Beau Britain
David Duffrin
Mackenzie Gray
Saad Usmani

New College of Florida
Distributed Computing for Data Science
Professor Lepinski

Summary

Background

- Complete Wikipedia edit history (up to January 2008)
- Very large file. 8 GB decompressing to $>$ 300 GB.
- Has 9 features, with Revision having sub-features.
  1. Revision: $article_{id}, rev_{id}, article_{title}, timestamp, [ip:]username, user_{id}$.
  2. Category: List of categories [Related tags].
  3. Images: List of images (each listed as many times as it occurs)
  4. Main: Cross-references to pages in other namespaces.
  5. External: Hyperlinks to pages outside Wikipedia.
  6. Template: List of all templates (each listed as many times as it occurs)
  7. Comment: Contains the comments as entered by the author.
  8. Minor: Whether the edit was marked as minor by the author.
  9. TextData: Word count of revision's plain text.

Exploring the data

Why is this even interesting?

- Did some exploratory analysis to find out what we would look at.
- Originally thought it would be interesting to look at page conflicts and see if we could identify edit wars.
- Deemed a bit out of our reach due to the data being somewhat sparse in that relation.
- Settled on edit diagnostics. The data is definitely there and we could tell a story about the edits.

Research Question

Research Question

Can we find any relationships between the users for the Wikipedia articles with the most number of edits?

Experimental Design

- Access TPT server
- Use Map-Reduce to subsample data
  - First get top 1,000 articles edited.
  - Get a sample of the dataset by getting all the records for those 1,000 articles.
- Using this sample, find all users that edited those articles.
  - Group by User, get edit count of each article.
  - Group by article, get vector of related tags.
- Use Principal Components Analysis (PCA)
- Create graphical network representations of data.

Design Limitations

- Troubles with accessing server and running jobs (slow fragmentation, disk space, random errors, Murphy's law).
- Large file. Had to use subsample. Couldn't use all of the data.
- The definition of "relationship"is limited to the related tags between articles and edits.
- Jobs are slow - and your code is never perfect. One error would take a couple hours later to fix.
- Analysis is mostly visual inspection.

Map-Reduce Functions

We will be using several map-reduce functions to subset our data and perform analysis on this Wikipedia edit database. The functions are as follows:

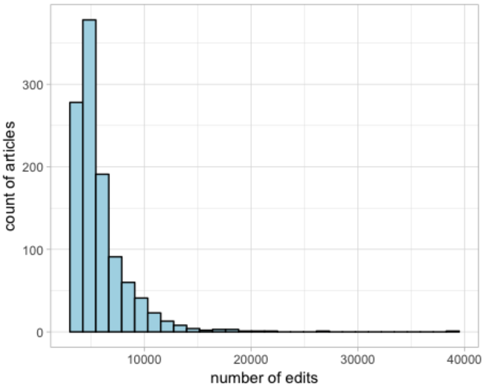
1. Getting the top edited articles.
2. Subset our data using the list of the top 1,000 edited articles.
3. Getting all the users who edited those articles.
4. Getting all the related tags of those articles.
5. Get the counts of all related tags in the records.

Top Articles Function

First, we used Map-Reduce to get the top thousand edited articles.

1. **Mapper**: Split the line, and extract the page name. Yield the key-value pair: $(pagename, 1)$ This gives 1 for every time a page name appears.
2. **Reducer**: Yield the page name and the sum of all 1s - the sum of every time the page is edited: $(pagename, sum(values))$

Histogram of Number of Edits



Subsetting Function

Next, we used Map-Reduce to subset the data using our 1,000 articles.

1. **Mapper**: From the entire dataset, split the line, and extract the page name. If the page name is in the list of the top articles extracted from the previous function, Yield the key-value pair: $(, (u_{id}, page_{name})$.
2. **Reducer**: Yield the entire record: (key, val)

User Function

Next, we can use this new subset and get the userids of each record.

1. **Mapper**: From the sample dataset, split the line, and extract the userid and article id. Yield the key/value pair: $[(user_{id} + +article_{id}), 1]$.
2. **Reducer**: Yield the sum of the times a user edited a specific article: $[key, sum(values)]$

From this, we created a python script to create a dataframe with a userid as an observation, and the list of 1,000 articles as features, and how many times a user edited that specific article.

Related Tags Function

Finally, we can get a vector of related tags for each article as well.

1. **Mapper**: From the sample dataset, split the line, and extract all of the related tags (categories). If the related tag is the list of top 1,000 articles, add them to a vector a related tags for that article. Yield the key/value pair: $[article_{id}, related - tags]$.
2. **Reducer**: Convert these related-tags into a set and yield the key/value pair: $[key, related]$

From this, we can create a preliminary network graph that could show relations between the user edits and the top edited articles.

Tag Count Function

Separate Map-Reduce function to get count of related tags.

1. **Mapper**: From the sample dataset, split the line, and extract all of the related tags (categories). Yield the key/value pair: $[tag, 1]$.
2. **Reducer**: Yield the tag and how many times it appears in all the records: $[key, sum(values)]$

This helps with some of the clustering we will be doing.

Results

For results, we will turn to an interactive module with: David Duffrin

References

1. J. Leskovec, D. Huttenlocher, J. Kleinberg. Governance in Social Media: A case study of the Wikipedia promotion process. AAAI International Conference on Weblogs and Social Media (ICWSM '10), 2010.
2. G. Kossinets. Processed Wikipedia Edit History. Stanford large network dataset collection.