Characterizing Topic Coverage on Wikipedia

Topical clustering of Wikipedia articles using the K-means Algorithms

Leo Chang

Dept. of Computer Science and Engineering

Santa Clara University

Santa Clara, CA

chihshun@gmail.com

Travis Ebesu

Dept. of Computer Science and Engineering

Santa Clara University

Santa Clara, CA

[tebesu@scu.edu](mailto:tebesu@scu.edu)

Stephi Hamilton

Dept. of Computer Science and Engineering

Santa Clara University

Santa Clara, CA

[s1hamilton@scu.edu](mailto:s1hamilton@scu.edu)

Bhavik Patel

Dept. of Computer Science and Engineering

Santa Clara University

Santa Clara, CA

[bpatel@scu.edu](mailto:bpatel@scu.edu)

Mohit Vadera

Dept. of Computer Science and Engineering

Santa Clara University

Santa Clara, CA

mvadera@scu.edu

*Abstract*— Wikipedia, “the free encyclopedia” has quickly become a comprehensive and authoritative source of information. This information is user generated, as users have the ability to submit new articles and edit existing articles on Wikipedia. As a result, topic coverage is user-driven. This project seeks to understand the coverage of topics in Wikipedia by clustering articles based on word frequencies. Using the Wikipedia dataset, we also compared the performance of the Mahout and WEKA implementations of the K-means algorithm.

*Index Terms*—K-means clustering, Mahout, WEKA, Wikipedia, Big Data Analytics

# Introduction

Wikipedia is a collaboratively edited, multilingual, free access content hosted by the Wikipedia Foundation. Articles are collaboratively edited and written by users, leading to the generation of 30 million articles, including 4.5 million English articles. Our work attempts to make a first step at characterizing the topic distribution of the English articles available on Wikipedia. This study attempts to make a first step in determining the variety of topics covered in Wikipedia, identifying areas with many articles as well as those with few articles.

Wikipedia has native category assignment to classify articles. Since the implementation of this feature in 2004, over 79,000 unique categories have been generated, all of which are implemented in a hierarchical tree structure, with categories and subcategories [1]. There are several problems with this system. Firstly, the articles are manually assigned to a category or subcategory, placing the onus of categorization on the author of the article. Because this process is manual with minimal standardization, it becomes difficult to validate the categorization of the article and determine the topic coverage of Wikipedia on the whole. Secondly, not every category available on Wikipedia has been properly rooted and placed in the hierarchical category structure available [1]. This leads to misclassification of articles, skewing their grouping

Knowing which topics are well covered, and which are not, will allow Wikipedia moderators and users to gain a better understanding of which areas need more articles and resources in order for Wikipedia to be a well-balanced, centralized resource for information on the internet. The clustering task that our project takes can also be a preliminary task for more complex tasks, such as evaluating the quality of articles, and recommending article topics for users to write about so that each topic has sufficient coverage.

In our experiments, we considered a raw dataset of 45 GB, which contained 4.5 million English articles. We then split the files by randomly choosing articles such that we had datasets of 32 MB, 1GB, and 5 GB. We then cleaned and vectorized the data so that each article was an observation, and the frequencies of each word present in the article were the features. We then applied the K-means algorithm on this vectorized data, where K represented the number of clusters available for the documents. In essence, the number of clusters was to represent the number of major topics covered by the articles. Therefore, a small number of clusters would represent a more granular grouping of articles, while a larger number of clusters would represent a more specific grouping of the articles. For this reason, we carried out the same experiments with different values of K.

In addition to varying the number of clusters, we carried out the clustering experiment on two different implementations of the K-means algorithm—Mahout [2], which runs on top of the Hadoop cluster, and WEKA [3], which runs on a single node. We ran the WEKA implementation of the algorithm on data sets of 32 MB. Additionally, we ran the Mahout implementation of the algorithm on datasets of 32 MB, 1 GB, and 5 GB. We compared performance of Mahout and WEKA on the small dataset, and evaluated the topic coverage on the large dataset.

# Related Work

Given the large size of the Wikipedia dataset and the free access to its entirety, it is no surprise that Wikipedia data is often used for K-means clustering studies. There are several ways that Wikipedia has found its way into machine learning and big data studies. Most often, Wikipedia data is used to augment the feature set, taking advantage of the linked-article structure to classify news articles [4]. However, for the purposes of our study, we decided on focusing on clustering Wikipedia articles only for the sake of simplicity.

Several attempts have been made to categorize the articles in Wikipedia by topic, each limited by data size or scope. One approach is to categorize articles is to simply look at the keywords associated with each article and cluster accordingly, reducing the number of features present in the dataset [5]. This relies on the keywords assigned to the article by the author, which does not guarantee to accurately represent the information found in the article, though the feature size considered is greatly reduced.  Others have attempted to visualize the semantic coverage of Wikipedia by visualizing the Wikipedia category network [1].

Our approach treats each article as a bag of words, without regard for the category network or assigned keywords. As mentioned previously, these are manually assigned, and are not guaranteed in accuracy. By treating each article as a bag of words, we can utilize the information in the whole article, expanding the number of features, which will allow us to capture more information. Since we do not want to depend on previously defined classifiers, we decided to use an unsupervised algorithm to classify articles with minimal manual intervention. Using the K-means algorithm is a straightforward approach to this problem, as each article can be vectorized, and the only manual input necessary is the value of K, or the number of clusters, for each experiment.

# Data Set Characteristics

The Wikipedia data dump is freely available at<http://dumps.wikimedia.org>. For this study, we utilized the latest data dump of English articles available on July 19, 2014, which was around 45GB total. After retrieving the data, we cleaned and split the data into three datasets by randomly choosing articles for each data set. We ended up with three separate data sets of 32 MB, 1GB, and 5 GB respectively. After this, the data was vectorized and utilized in the clustering implementations in WEKA and Mahout. Because of WEKA only runs on one node, we were only able to run the 32 MB dataset on it.

# Data Processing

The XML file available from the Wikipedia data dump was processed and cleaned before the K-means clustering algorithm could be applied to the data set. The total size of the raw data dump from Wikipedia was 45 GB. The data was cleaned using the Wikipedia parser found in the Mahout Project. Manual cleaning of the data was also necessary, which was accomplished using ad hoc scripts to remove special characters and other garbage characters from the dataset, and the Apache Lucene package was used to remove stop words and apply stemming. After data cleaning, the articles were partitioned into three data sets. The articles were randomly selected and placed into the three data sets, resulting in 32 MB, 1 GB, and 5 GB data sets.

## Mahout Data Cleaning

To clean and vectorize the data, we utilized several Mahout commands to further eliminate the HTML tags, URLs, and other special characters from the partitioned data. Additionally, the data was stemmed and stop words were removed. Then, the data was transformed to sequence files where the name of the article was the observation or row name. Mahout creates a dictionary of words to be used as features for each article. To calculate the value for each feature, the words in each article are tokenized, and the inverse-frequency (td-idf) is assigned as the value for each feature for an article. This minimizes the effect of common, high frequency words, allowing the clustering algorithm to weight less frequent significantly so that these features are used during clustering rather than common words. This yields a clustering result with articles classified according to significant words.

## WEKA Data Cleaning

In addition to the steps taken to process the data for the Mahout implementation, further steps were necessary to make the datasets compatible with WEKA. Separator and new line characters in the WEKA file format were removed, and the page HTML tags were replaced with new line characters. Then all redundant spaces and garbage characters were removed. Then each article was separated into two attributes, title and text. The title would be used to identify the article, and the text would be considered the “bag of words” for the clustering algorithm. The partitioned data was cleaned and stemmed. The data was then converted to .arff format for compatibility with WEKA. Using WEKA StringtoWord, the data was vectorized. To ensure important or significant words were used, the top 5000 words by frequency were included in the article sequence file.

# K-Means Algorithm Implementation

The K-means algorithm is a fairly simple and straightforward method of unsupervised learning. The algorithm requires an input of vectorized data, where each observation has values for a set of corresponding features. Then, a desired number of clusters, K, must be inputted to the algorithm. The algorithm then chooses K centroids at random and calculates the distance between the centroid and each observation. Each observation is assigned to the closest centroid. Each centroid is then recalculated to be the mean location of all the documents assigned to each centroid. This process is repeated until the within-cluster sum of squares is minimized [6].

The challenge with implementing the K-means algorithm is choosing an appropriate K. Given the fact that our study is exploratory, and we are dealing with a relatively large amount of data, we decided to go with the rule-of-thumb approach [8], which states:

We used this calculation as a starting point for choosing different values of K.

## WEKA Implementation

To cluster the data, the SimpleKMeans algorithm provided by WEKA was used. To run the algorithm, the number of clusters was defined. The seeding method was chosen as random, which means the initial means for the K-clusters would be randomly chosen by the algorithm. The Euclidian distance was used to calculate the distance between centroids and documents and to recalculate the centroids for each iteration of the algorithm.

The WEKA implementation of K-means was run on a Intel Core i5 4570R 4-Cores machine with 1Gbit bandwidth. The machine has 8GB DDR3 RAM. A maximum heap size of 2 GB was configured for WEKA, and WEKA’s performance was not affected when heap size was increased.

## Mahout Implementation

Mahout implements the K-means algorithm within the MapReduce paradigm [7]. Here, the data and the K centroids are treated as key-value pairs. The keys are the document title or cluster identification, and the values are the frequencies of each word or the location of the centroid for the current iteration of the algorithm. In the map step, all cluster centroids are read into memory from the sequence file. Then the algorithm iterates over each cluster center for each document key/value pair. The algorithm measures the distance between each document vector and each cluster vector, saving the center closest to each document. The cluster centers, with a vector of documents assigned to each, are written to the file system. In the reduce step, the algorithm iterates over each document assigned to each cluster and calculates the average value. This value becomes the new centroid for the cluster and saved to a sequence file. Then the newly calculated centroid and the centroid stored in the key object are compared. If the difference between the two is below a certain threshold or the maximum number of iterations has been reached, the algorithm ends; however, if the difference between the two is above the threshold, another iteration of the K-means algorithm is run, unless the limit on iterations has been reached. A maximum of five iterations was set so that a result could be generated in a reasonable amount of time; otherwise, the algorithm was having difficulty converging to a particular solution. This is reasonable, as we are processing many documents, and our vectors represent high dimensional data.

The Mahout implementation was run on top of a Hadoop cluster with 48 Map slots and 48 Reduce slots. There were 24X Worker nodes, including the Data nodes and Task Tracker, each with 2 Cores, 4GB RAM and a bandwidth of 1Gbit.

# Results

## Execution Time Comparison

We used the Wikipedia dataset to benchmark the running times of Mahout (implemented on Hadoop) and WEKA (implemented on a single node). Due to memory limitations, we were only able to run the 32 MB file on both WEKA and Mahout. In addition, we were able to successfully complete the K-means calculations on 1GB and 5GB files using Mahout on Hadoop. The results are shown in Table 1 and Table 2.

**Table 1:** Execution Times for WEKA implementation

|  |  |  |
| --- | --- | --- |
| **Data Set Size** | **K -Value** | **Time** |
|  | 50 | 2.52 minutes |
| 75 | 3.87 minutes |
| 100 | 3.68 minutes |

**Table 2:** Execution Times for Mahout Implementation

|  |  |  |
| --- | --- | --- |
| **Data Set Size** | **K-Value** | **Time** |
| 32 MB | 50 | 2.22 minutes |
| 75 | 2.27 minutes |
| 100 | 2.65 minutes |
| 1 GB | 500 | 44.35 minutes |
| 1000 | 96.18 minutes |
| 2000 | 113.18 minutes |
| 5 GB | 500 | 51.6 minutes |
| 1000 | 83.8 minutes |
| 2000 | 124.72 minutes |

## Clustering Result Evaluation

To evaluate the clustering algorithm, we looked at the radii of each cluster for each value of K. Specifically, we looked at the minimum and maximum radii and the standard deviation of the radii for each value of K evaluated to get an idea of the size and spread of each of the clusters. We decided to look at the larger datasets (1 GB and 5 GB) because these would most closely simulate the scale and size of the data if we were to analyze the entire Wikipedia repository.

According to our chosen metric, the best clustering result came from the 5 GB data set, where the maximum radius and standard deviation were smallest. Given the K value of 1000, we can assume that there were 1000 topics covered in the collection of articles in the 5 GB data set.

**Table 3:** Cluster Analysis of Mahout implementation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Set Size** | **K -Value** | **Radii Min** | **Radii Max** | **Radius Standard Deviation** |
| 1 GB | 500 | 0 | 133659.066 | 13183.735 |
|  | 1000 | 0 | 449903.418 | 34301.468 |
|  | 2000 | 0 | 953991.108 | 214127.937 |
| 5 GB | 500 | 0 | 230299.742 | 46912.584 |
|  | **1000** | **0** | **500.000** | **105.160** |
|  | 2000 | 0 | 970030.168 | 92427.909 |

All distances measured using Euclidean distance

# Conclusions And Future Work

## Conclusions

Looking at the execution time data for the 32 MB data set, the Mahout and WEKA implementations run comparably for a small number of clusters. However, WEKA does not scale well when evaluating the algorithm for larger values of K. Despite the overhead associated with a distributed file system, Hadoop still beats WEKA in execution time. This is probably because of the high-dimensional nature of the data, which Hadoop is better designed to handle.

Given the clustering analysis results, the best clustering occurred in the 5 GB data set with a K value of 1000. An optimal value for the 1 GB data set was not found, but further analysis should indicate an optimal K value between 500 and 1000. The 5 GB data set represent a small subset of the entire Wikipedia dataset, but the results are promising, showing that it is possible to cluster the articles using an unsupervised learning technique, where the clusters roughly match up with article topics.

## Evaluation of WEKA and Mahout

WEKA is a relatively fast and efficient tool to use for small datasets because it runs the K-means algorithm on a single node. However, it is limited by the memory specifications of the machine on which it is running. There is no replication factor to protect against failing, but because there is only one node running the algorithm, there is little time lost for setup. However, WEKA has many built-in filters and features that make it easy to work with data. For instance, WEKA automatically vectorizes categorical data, and when the data is sparse, it applies filtering so that the data is compressed further, increasing the amount of data that can be processed.

In contrast, Hadoop and Mahout are built for large-scale data processing. Hadoop uses HDFS 64 MB block size and a minimum replication factor of three to prevent failure. The distributed file system and MapReduce paradigm allow large amounts of data to be processed efficiently. However, there is an overhead associated with setup and maintenance of this distributed system. Hadoop is also designed for redundancy so that if one node fails, another node can process the same job, minimizing errors and delays. Despite the safeguards in place preventing the failure of a single node from impacting the whole job, there is an improbable, but not impossible, source of single node failure in Hadoop--the NameNode, which houses the directory tree of all files in the file system. If this node fails, the entire system will fail.

Eventually, it would be ideal to process the entire Wikipedia dataset. Hadoop would be best suited for that purpose over WEKA due to its ability to process large datasets. Future work might include finding a robust solution to the single point of potential failure found in the NameNode, which may incorporate the distributed NameNode system found in MapR’s Hadoop implementation or a standby NameNode found in Cloudera’s implementation. Also, it would be beneficial to reduce the execution time of large jobs on the Hadoop cluster. Implementing solutions like Apache Spark could result in running programs 100x faster than Hadoop MapReduce in memory. Finally, the current setup of the cluster is inadequate and contains a subpar configuration. The number of mappers cannot be specified. On the 1GB dataset only 3 mappers run, on a 5GB data set 17 mappers run. Thus the time elapsed to process is nowhere near the five times size in dataset. Furthermore, inadequate memory and heap size allocation was often a problem. 7GB was the maximum memory that could be allocated to a JVM, which still produced an occasional memory error.

## Future Work

In addition to upgrades to the Hadoop cluster to accommodate the large scale of this kind of project, work can be done in refining the clustering result. In our project, we used the K-means algorithm to cluster the data. Variations of this algorithm, such as K-means++, where the initial centroids are chosen so the algorithm runs faster, and K-medioids, which has a more robust method of choosing the centroid, could be used to refine the result. Additionally, using techniques such as fast search and choosing centers according to density will eliminate the need to set the K value and detect outliers more efficiently.

Additionally, characterizing and vectorizing the data differently may lead to a more accurate clustering result. Machine learning techniques, such as PCA, can be used to reduce the dimensionality of the data, making it easier to process, run, and understand. Our approach treated each article as a bag of words, with disregards the context of the words within the article. Techniques in context-aware clustering, most often implemented in a semi-unsupervised learning form, can allow for the context of specific words to play a role in the weight they are given as features in the clustering algorithm.

This preliminary work on clustering documents by topic could be used to improve the current categorization strategy used on Wikipedia. Instead of depending on manual classification, clustering could be used to build a model where documents can be classified based on the cluster in which they best fit. Additionally, these analyses could be used to direct Wikipedia curators in managing the information present, and finding ways to recruit users to write articles on topics where information may be lacking.

# References

1. T. Holloway, M. Bozicevic, and K. B¨orner. Analyzing and visualizing the semantic coverage of Wikipedia and its authors: Research articles. Complexity, 12(3):30–40, 2007.
2. http://mahout.apache.org/
3. [http://www.cs.waikato.ac.nz/ml/WEKA/](http://www.cs.waikato.ac.nz/ml/weka/)
4. Gabrilovich, E. and Markovitch, S. (2007) Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In Proceedings of the Twenty-First National Conference on Artificial Intelligence, Boston, MA.
5. C. Wartena and R. Brussee. Topic detection by clustering keywords. In DEXA ’08, pages 54–58, 2008.
6. J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297, Berkeley, CA, 1967. University of California Press.
7. J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in Proceedings of OSDI ’04: 6th Symposium on Operating System Design and Implementation, San Francisco, CA, Dec. 2004.
8. [Kanti Mardia](http://en.wikipedia.org/wiki/Kanti_Mardia) et al. (1979). *Multivariate Analysis*. Academic Press.