Comment Analysis: Deep Dive Into Wikipedia Data

Adhok, Katti (N10548394)

IFN704 Assessment 3: Final report

Executive Summary

The Wikimedia Foundation makes all of Wikipedia's data available in XML format and most of the machine learning Natural Language-Processing (NLP) tasks, such as word sense disambiguation and semantic relatedness measures are now used on the said data for database mining. However, there was no work found that was done on the analysis of user comments on the Wikipedia page. Hence, this study aims to implement NLP on the comments of Wikipedia data extracted from a High Performance Compute (HPC) at the Queensland University of Technology and attempts a nontrivial extraction of implicit, previously unknown, and potentially beneficial information from data.

To begin, the HPC was used to query the required Wikipedia data on page comments and its related features on a few of Europe's successful football clubs. Second, word tokenization on the data was performed using a new and advanced NLP library, namely SpaCy. Finally, an unsupervised machine-learning algorithm was implemented on the processed data for extracting useful insights about Wikipedia comments.

1 Introduction

Knowledge discovery is the finding of implicit, previously unknown, and potentially helpful information from data. Experiments that are designed to decrease the impact of all except a few parameters and emphasize the change of one or a few target parameters generate scientific data. Typical corporate databases, on the other hand, collect a lot of information about their users to serve a range of organizational needs. Frawley et al. [1] in their journal have described that knowledge discovery in databases should have four distinct properties: High-Level language, accuracy, interesting results and efficiency. This research aims to query a large dataset stored on the Queensland University of Technology's high-performing computer for extraction of comments from specific pages of Wikipedia. Computational (and/or data visualization) resources and services used in this work were provided by the eResearch Office, Queensland University of Technology, Brisbane, Australia. This is then followed by implementing a machine learning algorithm with the goal of knowledge discovery and acquiring useful insights into the world's biggest encyclopedia's user behavior. We also examine the shortcomings discovered throughout the study and offer potential improvements and future work for the research.

Wikipedia, the online and free encyclopedia, has become one of the most major and comprehensive information sources during the last decade. It has evolved into a phenomenon among academic researchers as well as the general public, and there has been a large influx of scientific research and studies that have been applied to Wikipedia datasets to address a variety of challenges in recent years. Medelyan et al. [1] provide a comprehensive review of a variety of natural language processing (NLP) algorithms used for information mining from Wikipedia data. Their research is based on the idea that Wikipedia, with its unique combination of scale and structure, bridges the gap between quality and quantity. As a result of their research, it is safe to conclude that Wikipedia is a good candidate for possible NLP applications, as well as an information resource with astonishing depth and use in today's ever-expanding knowledge base. Hence, with this research opportunity, an attempt was made to apply NLP to the user comments of the Wikipedia dataset. Since clustering is an effective method for extracting categories from a text on a large scale [3], K-means clustering will be used the cluster the comments in the data and analyze the cluster results.

Before further discussion, a brief introduction to Wikipedia's XML schema for its data is discussed. Wikipedia's XML file structure is composed of only two major object types: the *siteinfo* object and the *page* object. Each of these objects has several associated sub-objects and fields.

The *siteinfo* object, as shown in Figure 1, is made up of the following elements:

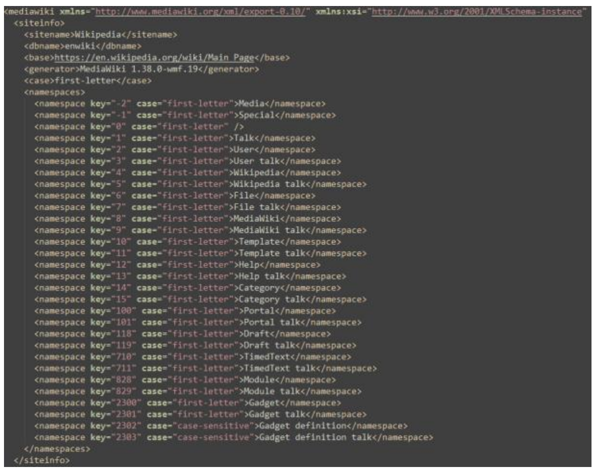
1.1 *dbname*: a unique identifier for this Wikipedia instance.

1.2 *sitename*: the title of the Wikipedia that may be shown.

1.3 *base*: the primary link to Wikipedia's base page.

1.4 *generator*: details about the MediaWiki software at the time the data was created.

1.5 *case*: sitename formatting to denote the position of the uppercase letter.



```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.10/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <dbname>enwiki</dbname>
    <base>https://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.38.0-wmf.19</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Special</namespace>
      <namespace key="0" case="first-letter" />
      <namespace key="1" case="first-letter">Talk</namespace>
      <namespace key="2" case="first-letter">User</namespace>
      <namespace key="3" case="first-letter">User talk</namespace>
      <namespace key="4" case="first-letter">Wikipedia</namespace>
      <namespace key="5" case="first-letter">Wikipedia talk</namespace>
      <namespace key="6" case="first-letter">File</namespace>
      <namespace key="7" case="first-letter">File talk</namespace>
      <namespace key="8" case="first-letter">MediaWiki</namespace>
      <namespace key="9" case="first-letter">MediaWiki talk</namespace>
      <namespace key="10" case="first-letter">Template</namespace>
      <namespace key="11" case="first-letter">Template talk</namespace>
      <namespace key="12" case="first-letter">Help</namespace>
      <namespace key="13" case="first-letter">Help talk</namespace>
      <namespace key="14" case="first-letter">Category</namespace>
      <namespace key="15" case="first-letter">Category talk</namespace>
      <namespace key="100" case="first-letter">Portal</namespace>
      <namespace key="101" case="first-letter">Portal talk</namespace>
      <namespace key="118" case="first-letter">Draft</namespace>
      <namespace key="119" case="first-letter">Draft talk</namespace>
      <namespace key="710" case="first-letter">TimedText</namespace>
      <namespace key="711" case="first-letter">TimedText talk</namespace>
      <namespace key="828" case="first-letter">Module</namespace>
      <namespace key="829" case="first-letter">Module talk</namespace>
      <namespace key="2300" case="first-letter">Gadget</namespace>
      <namespace key="2301" case="first-letter">Gadget talk</namespace>
      <namespace key="2302" case="case-sensitive">Gadget definition</namespace>
      <namespace key="2303" case="case-sensitive">Gadget definition talk</namespace>
    </namespaces>
  </siteinfo>
```

Figure 1. XML schema of the *siteinfo* object. The major element of this structure contains information about the source of the data, notably a link to the wiki's home page and the sitename.

The *page* object, as shown in Figure 2, is made up of:

1.1 *title*: the well-formatted title of the page.

1.2 *ns*: refers to the namespace of the page.

1.3 id: unique identification number of the page.

1.4 *revision*: child object of *page*. The *revision* object contains:

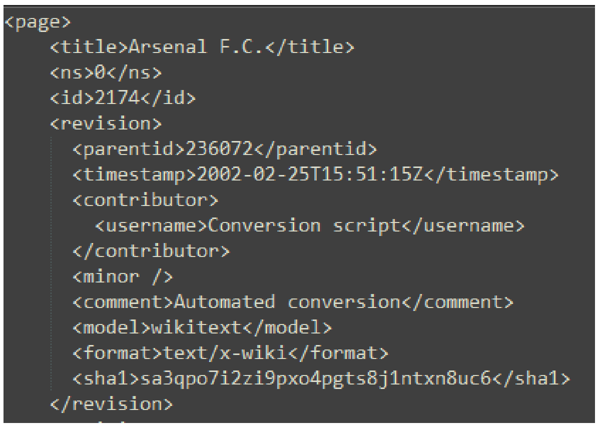
1.4.1 *parentid*: the id of the previous revision before this one for the same page.

1.4.2 *timestamp*: the date and time when this revision was created.

1.4.3 *comment*: remarks added by the person that has provided this revision.

1.4.4 *model*: the model used to format this text.

1.4.5 *text*: the text embedded in the Wikipedia page (not seen in Figure 2).



```
<page>
    <title>Arsenal F.C.</title>
    <ns>0</ns>
    <id>2174</id>
    <revision>
      <parentid>236072</parentid>
      <timestamp>2002-02-25T15:51:15Z</timestamp>
      <contributor>
        <username>Conversion script</username>
      </contributor>
      <minor />
      <comment>Automated conversion</comment>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <sha1>sa3qpo7i2zi9pxo4pgts8j1ntxn8uc6</sha1>
    </revision>
```

Figure 2. XML schema of the *page* object. This object contains all of the information for a particular page present within the large XML file.

Yasseri et al. [2] delineate in their research on how sports clubs and athletic events are among the top categories receiving contentious Wikipedia page edits from their supporters. However, this research has focused on comment analysis on just one football club from the English Premier League just out of sheer love and passion for sports, namely Manchester United F.C.

2 Literature Review

There is a wide range of ongoing research on knowledge discovery in databases. Let us begin by defining and discussing a few crucial terms As previously mentioned, Medelyan et al. [1] infer that Wikipedia has now turned into a popular study topic in the field of computer science and there is a growing number of studies regarding the use of Wikipedia to improve text mining tasks.

Banerjee et al. [3] clusters feed readers comparable articles to make the information better accessible for its user. They utilize the open-source clustering package SenseClusters2 which uses CLUTO3 as its clustering engine and offers six distinct clustering algorithms in that one package. Their main idea is to cluster short text items from Wikipedia for knowledge discovery. The results obtained by them were found to corroborate recent discoveries made on Wikipedia's information retrieval methods. But their research focused on the RSS feed of Wikipedia. Inspiration for this research, to clustering comments after NLP processing, was drawn from Banerjee's research.

Another text-based document clustering was researched by Sedding, J [6] that attempted to cluster text documents according to semantic similarities using WordNet. The overarching goal of Sedding's study was to cluster *Reuters-21578* documents according to their meanings. That is, to include synonyms, hypernyms and disambiguated words for English part-of-speech tagging to improve clustering effectiveness. Though Sedding's work was vastly different from what is tried in this study, the concept of corpus processing was acquired and utilized in this work.

Both Wang, P., & Domeniconi, C [7] and Hu, Jian, et al. [8] similarly used category information from Wikipedia text for clustering. Their approaches make extensive use of Wikipedia's enormous structural information. They both, however, use an exact phrase-matching technique to link the text documents and further cluster and classify them respectively. This method is constrained by the phrases used in the documents as well as the coverage of Wikipedia articles for the analysis.

Gabrilovich, E. [9] present an approach for incorporating text classification into Wikipedia. They essentially create an auxiliary text classifier which can match documents with the most relevant Wikipedia articles. Thereafter, they create and enrich a bag-of-words model with additional features described by related Wikipedia articles. This feature-generating approach works in the same way as a data retrieval process does. It takes input as a text fragment that is generated earlier but then maps it onto the most appropriate Wikipedia pages.

Numerous research has been conducted on Wikipedia articles with various machine-learning approaches. Most of the corpus analysis of Wikipedia data is used to detect user emotions towards positive and negative qualities of the articles. Clustering is widely used with WordNet is widely used in ontologies for text-based classification of massive article collections. However, there were no works found that implemented NLP as a preprocessor to the text corpus of Wikipedia data and subsequently clustered the text corpus. As a result, an attempt was made to do language processing and clustering on the comments section of Wikipedia data.

3 Methodology

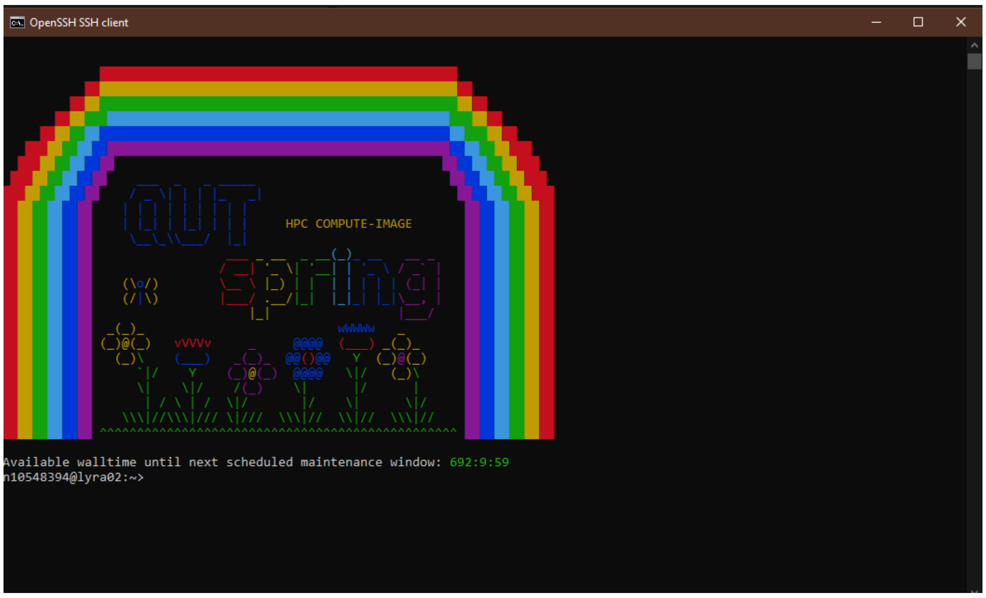
Because of the magnitude of the whole Wikipedia dataset, doing a full-scale analysis throughout all the articles and including all the comments, revisions, and users is quite a challenging task, hence most research in this study only utilizes a small portion of the data. The machine-learning algorithm's implementation receives a lot of attention in this study.

A complete data dump of Wikipedia articles in the English language was made available on the HPC during the months of April-May 2022 and access to the said HPC was granted upon request and nature of use. The data consists of over 70 million Wikipedia pages that are encyclopedic article-related information, totaling nearly 198 gigabytes of data. These 70 million Wikipedia pages exist distributed as a set of 7zip archive files, each representing a single XML file. There are 743 compressed files, separated based on their page numbers. To handle data of this size and scale, queries were submitted to the HPC in python language to extract data from the articles pertaining to the comments from the *comment* object. Initial code development is borrowed from Dimitri Perrin's [10] work published on GitHub. The code helped in locating and unzipping a specific archived file(s), extracting data for the page of interest from the archived files, stripping the XML file(s) of all the *text* objects so that the file size diminishes and finally saving the data to a new XML file. The resulting file achieved from the HPC was merely a fraction of its original file size, with only the required information.

3.1 HPC (High Performance Compute)

The Queensland University of Technology's supercomputer or HPC is available to all its research students and faculty and is fondly known as Lyra. It is a computing cluster made up of CPU 146 nodes, CPU 7640 cores, 68TB of RAM and 73 various Nvidia GPUs on a highly scalable HPE Data Management Framework 7 that manages a whopping 11 PB of data [11].

As supplemental information, all Python task scripts submitted to the HPC are attached. The code may be found in the Appendix.

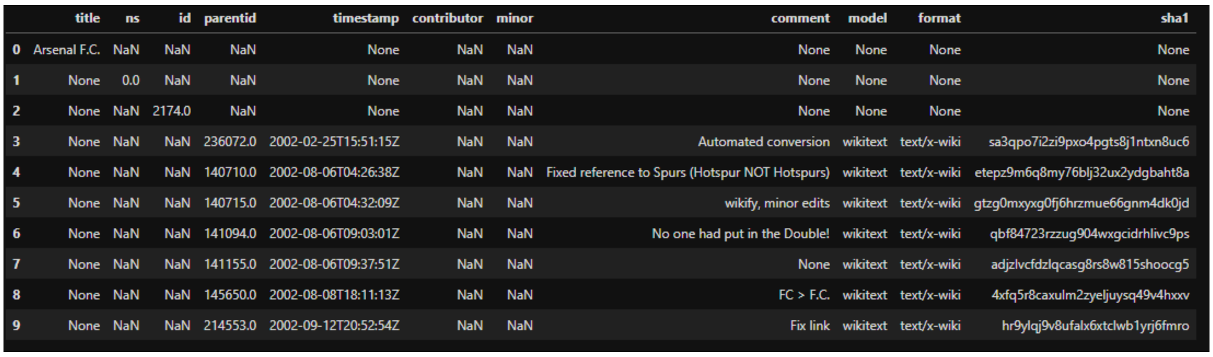


```
Available walltime until next scheduled maintenance window: 692:9:59
n10548394@lyra02:~>
```

Figure 3: Snippet of Lyra, as run on Windows Command Prompt, where jobs are submitted.

3.2 Approach

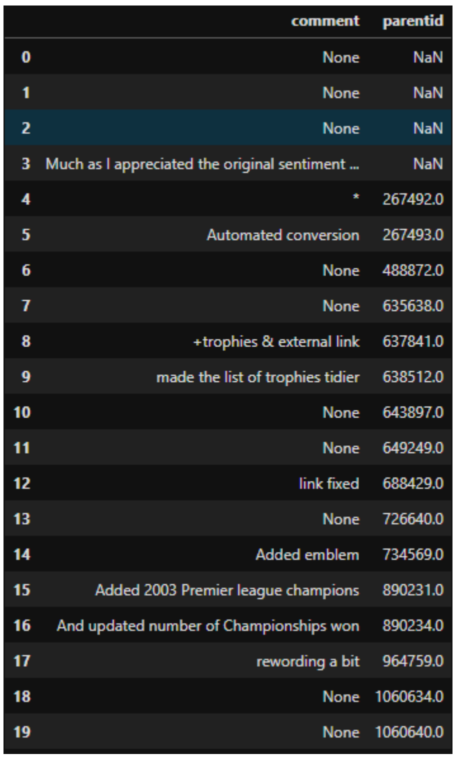
The data extracted from HPC was then analyzed locally using python as the preferred programming language and JupyterLab as the GUI for data analysis. The following table describes the data that was extracted from the HPC:



	title	ns	id	parentid	timestamp	contributor	minor	comment	model	format	sha1
0	Arsenal F.C.	NaN	NaN	NaN	None	NaN	NaN	None	None	None	None
1	None	0.0	NaN	NaN	None	NaN	NaN	None	None	None	None
2	None	NaN	2174.0	NaN	None	NaN	NaN	None	None	None	None
3	None	NaN	NaN	236072.0	2002-02-25T15:51:15Z	NaN	NaN	Automated conversion	wikitext	text/x-wiki	sa3qpo7i2zi9pxo4pgts8j1ntxn8uc6
4	None	NaN	NaN	140710.0	2002-08-06T04:26:38Z	NaN	NaN	Fixed reference to Spurs (Hotspur NOT Hotspurs)	wikitext	text/x-wiki	etepz9m6q8my76blj32ux2ydgbaht8a
5	None	NaN	NaN	140715.0	2002-08-06T04:32:09Z	NaN	NaN	wikify, minor edits	wikitext	text/x-wiki	gtzg0mxyxg0fj6hrzmue66gnm4dk0jd
6	None	NaN	NaN	141094.0	2002-08-06T09:03:01Z	NaN	NaN	No one had put in the Double!	wikitext	text/x-wiki	qbf84723rzzug904wxgcidrhlivc9ps
7	None	NaN	NaN	141155.0	2002-08-06T09:37:51Z	NaN	NaN	None	wikitext	text/x-wiki	adjzlvcfdzlqcasg8rs8w815shoocg5
8	None	NaN	NaN	145650.0	2002-08-08T18:11:13Z	NaN	NaN	FC > F.C.	wikitext	text/x-wiki	4xfq5r8caxulm2zyeljuysq49v4hxxv
9	None	NaN	NaN	214553.0	2002-09-12T20:52:54Z	NaN	NaN	Fix link	wikitext	text/x-wiki	hr9ylqj9v8ufalx6xtclwb1yrj6fmro

Table 1: XML data extracted from the HPC.

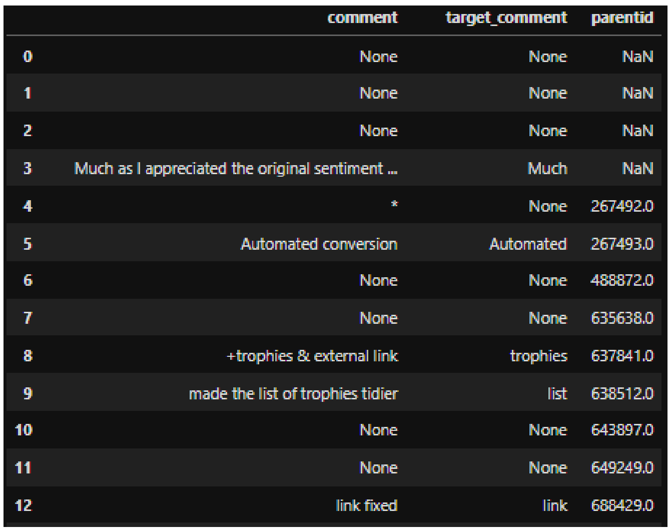
The Pandas library was used to read the XML file, and Figure 4 displays the XML file as prepared by the Pandas library, data frame's columns are detailed in the introductory section. Since a lot of text was stripped from the XML files, this approach was found to be best suited for importing the XML document. Columns *title*, *ns*, *id*, *model*, *format* and *minor* were found to have only one value as they were the metadata of just one page from Wikipedia. Hence they were disregarded from the analysis. Columns *timestamp*, *sha1*, and *contributor* were dropped to reduce the complexity of the model.



	comment	parentid
0	None	NaN
1	None	NaN
2	None	NaN
3	Much as I appreciated the original sentiment ...	NaN
4	*	267492.0
5	Automated conversion	267493.0
6	None	488872.0
7	None	635638.0
8	+trophies & external link	637841.0
9	made the list of trophies tidier	638512.0
10	None	643897.0
11	None	649249.0
12	link fixed	688429.0
13	None	726640.0
14	Added emblem	734569.0
15	Added 2003 Premier league champions	890231.0
16	And updated number of Championships won	890234.0
17	rewording a bit	964759.0
18	None	1060634.0
19	None	1060640.0

Table 2: Data frame used for the analysis of comments.

Column *parentid* was incorporated in the data as it maps to the comment made by the user. Thereafter, the comment column, which is a string, was tokenized using the SpaCY library. All the punctuation, stop words and digits are removed from the *comment* column and the resulting data is shown as follows:



	comment	target_comment	parentid
0	None	None	NaN
1	None	None	NaN
2	None	None	NaN
3	Much as I appreciated the original sentiment ...	Much	NaN
4	*	None	267492.0
5	Automated conversion	Automated	267493.0
6	None	None	488872.0
7	None	None	635638.0
8	+trophies & external link	trophies	637841.0
9	made the list of trophies tidier	list	638512.0
10	None	None	643897.0
11	None	None	649249.0
12	link fixed	link	688429.0

Table 3: Data frame of the target *comment* column after tokenization and keeping only the keywords from the string of sentence.

The target variable named *target_comment*, as shown in Table 3, has short texts that are wisely chosen by the SpaCy library and are kept consistent throughout the variable. This target column is then label-encoded to make the target variable suitable for clustering and as the last step, the data is standardized before performing clustering.

A k-means clustering algorithm was used to cluster the variables due to its easier convergence and the elbow method was used to evaluate the optimal centroids for the data. Silhouette score is a measure of a cluster's quality [12] and it calculates an average representative score based on the closeness of individual clusters (intra-cluster distance) to the distance of clusters as a whole (inter-cluster distance). The Calinski-Harabasz (CH) score [13] is most typically used to assess the quality of goodness of split of a K-Means clustering algorithm with a certain number of clusters. Hence, these two scores were imparted to measure and evaluate the clusters obtained from the analysis. A Silhouette score has a value ranging from -1 to 1. A score of 1 means that the clusters are neatly separated and easily identifiable and -1 indicates that clusters have been assigned incorrectly. A Silhouette score of 0 indicates that the clusters are generally indifferent, or that the separation between the obtained clusters is insignificant. Similarly, for all clusters, the Calinski-Harabasz index is derived as a ratio of the summation of inter-cluster dispersion and the sum of intra-cluster deviation. A high CH score indicates improved clustering as observations in each cluster are nearer to each other and the clusters themselves are wider apart with good separation.

4 Findings

From the elbow method, it was determined that 8 clusters are optimal for the data.

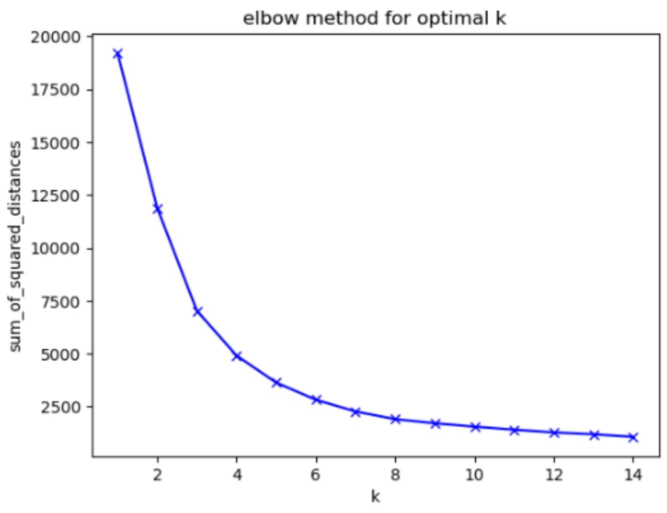


Figure 4: Elbow method for determining the optimal value of cluster.

Number of Clusters, k	Silhouette score	CH score
6	0.4770	11149.8394
8	0.5048	12452.8429
10	0.4511	12036.200

Table 4: Statistics for various clusters k.

By looking at the elbow graph, as shown in figure 4, and the statistics for a few initial clusters, it was determined that 8 clusters would be initialized for the data to cluster comments that are similar in nature.

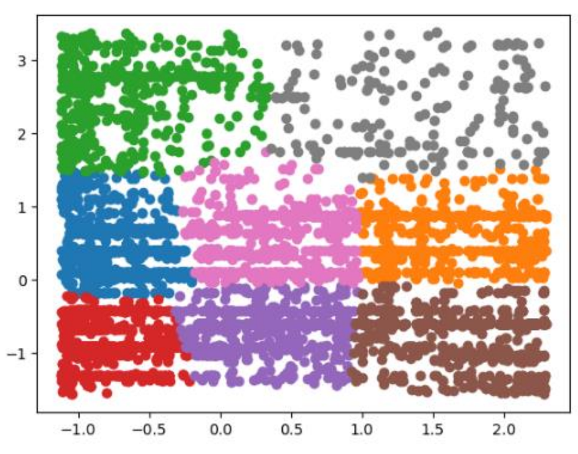


Figure 5: Clustered short text comments.

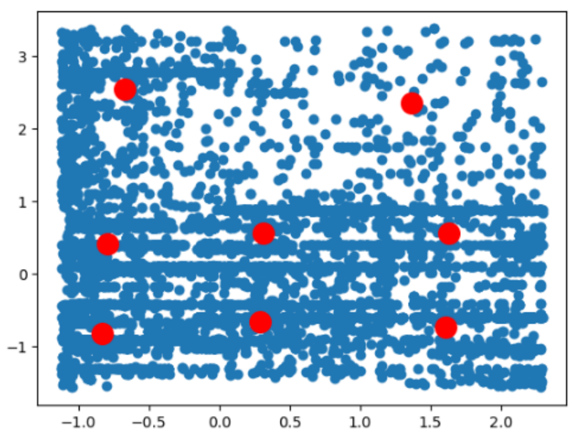


Figure 6: Centroids of the clustered comments.

We observe that the clusters are well separated and centroids are distinct from each other. The approach implemented in this study has resulted in comments clustered not overlapping as parentid provides each datapoint a unique value. Further discussion about the clusters, limitations in the analysis and possible future works are discussed in the next section.

5 Reflection

Because there has been no previous work on the user comments of the Wikipedia dataset, this study aims to provide a first proposal and implementation pipeline for using NLP in the comment analysis of the data. Clearly, there are a few limitations that were discovered during the lifetime of this research.

5.1 Word Tokenization

At the time of word tokenization, one limitation of the study was found. Once the comments are tokenized, SpaCy has shown large deviations in selecting the verb from the strings of comments. This is a characteristic property of SpaCy and other NLP libraries like NLTK and Gensim can be implemented and explored.

5.2 Generalization

The implementation and analysis of the NLP and a machine-learning algorithm on this dataset have been restricted to only one page. More work on a wide range of Wikipedia articles will certainly help in producing a robust model that can be used for knowledge discovery.

5.3 Cluster Analysis

This research aimed to discover valuable insights from the Wikipedia dataset using NLP and machine learning algorithms. However, at the conclusion of the report, the clusters of the user comment that are obtained by the data model have to be still analyzed in depth. On the overview of the clustered comments, it is found that the clusters are separated in the desired manner but the contents of the clusters themselves are found to be quite dissimilar. The clusters elements of each cluster are not homogeneous in nature and have words that are not similar to each other. Further deep dive into increasing the number of clusters or performing parameter tuning on the dataset can help achieve a better score.

To summarise and conclude the findings of this research, it can be concluded that performing NLP on the Wikipedia dataset can most certainly help in better analysis of the semantics and nature of comments themselves, however, better machine-learning algorithms can be implemented for knowledge discovery from the data. As Frawley et al. [1] mentioned, the results of this study on implementing NLP and clustering can only attribute to two parts of the successful knowledge discovery process, which are using a high-level language for the analysis and discovering a few interesting facts from the dataset. However, more work has to be put into the research to make the study more accurate and efficient.

6 References

[1] W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, “Knowledge Discovery in Databases: An Overview,” AI Magazine, vol. 13, 1992.

[2] O. Medelyan, D. Milne, C. Legg and I. H, “Mining meaning from Wikipedia,” International Journal of Human-Computer Studies , p. 716–754, 23 February 2009.

[3] B. Larsen and C. Aone, “Fast and Effective Text Mining Using Linear-time Document Clustering,” Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 16-22, 1999, August.

[4] T. Yasseri, A. Spoerri, M. Graham and J. Kertesz, “The Most Controversial Topics in Wikipedia,” Global Wikipedia: International and cross-cultural issues in online collaboration, pp. 25 - 48, 2014 .

[5] S. Banerjee, K. Ramanathan and A. Gupta, “Clustering Short Texts using Wikipedia,” Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 787-788, 2007.

[6] J. Sedding and D. Kazakov, “Wordnet-based text document clustering,” Proceedings of the 3rd workshop on RObust Methods in Analysis of Natural Language Data, pp. 104-113, 2004.

[7] P. Wang and C. Domeniconi, “Building Semantic Kernels for Text Classification using Wikipedia,” Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 713-721, 2008, August.

[8] J. Hu, H.-J. Zeng, H. Li, Z. Chen, L. Fang, Y. Cao and Q. Yang, “Enhancing Text Clustering by Leveraging Wikipedia Semantics,” Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 713-721, 2008.

[9] E. Gabrilovich and S. Markovitch, “Feature Generation for Text Categorization Using World Knowledge,” IJCAI, vol. 5, pp. 1048-1053, 2005.

[10] D. Perrin, “SystemsResearch: Wikidata,” https://github.com/, Brisbane, 2022.

[11] Queensland University of Technology , “About QUT's supercomputer,” Queensland University of Technology, 2022. [Online]. Available: https://qutvirtual4.qut.edu.au/group/research-students/conducting-research/specialty-research-facilities/advanced-research-computing-storage/supercomputing.

[12] K. R. Shahapure and C. Nicholas, “Cluster Quality Analysis Using Silhouette Score,” International Conference on Data Science and Advanced Analytics, pp. 747-748, 2020.

[13] C. Tadeusz and J. Harabasz, “A dendrite method for cluster analysis,” Communications in Statistics-theory and Methods , vol. 3, no. 1, pp. 1-27, 1974.

7 Appendix

7.1 Unzip and extract data from HPC

Untitled2

November 8, 2022

```
[ ]: import glob, subprocess, datetime, ast
     from bs4 import BeautifulSoup as bs

     folder_in = "/work/wikidata/n10548394/"
     folder_out = "/work/wikidata/n10548394/"

     files = ["enwiki-20220201-pages-meta-history9.xml-p3428722p3464864.notext"]

     def progress_message(txt):

         # datetime object containing current date and time
         now = datetime.datetime.now()
         # time, formatted as dd/mm/YY H:M:S
         dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
         # we print the time and the message
         print(dt_string,txt)



     def extract_data(filename, folder):

         with open(filename, "r") as inFile:
             content = inFile.readlines()
             content = "".join(content)
             bs_content = bs(content, "lxml")

         progress_message(f"Data in memory. Parsing.")

         pages = bs_content.find_all("page")
         progress_message(f"{len(pages)} pages to process")

         nb_page_processed = 0

         temp_name = filename.split("/")[-1]
         output_name = temp_name+".csv"

         with open(folder+output_name,'w') as outFile:
```

```
      outFile.write("Page ID,Page title,Revision ID,Timestamp,Contributor␣
↪ID,Contributor name\n")

      for p in pages:

          # we extract the page title
          title = p.find("title")
          title_value = str(title)[7:-8]

          # we extract the page ID
          ID = p.find("id")
          ID_value = str(ID)[4:-5]

          # we extract the revisions
          revisions = p.find_all("revision")

          # for each revision
          for r in revisions:

              # we extract the revision ID
              rev_ID = r.find("id")
              rev_ID_value = str(rev_ID)[4:-5]

              # we extract the timestamp
              rev_time = r.find("timestamp")
              rev_time_value = str(rev_time)[11:-12]

              # we extract the contributor
              contributor = r.find("contributor")

              # the contributor can be a registered user or an IP
              contributor_ID = contributor.find("id")
              if contributor_ID != None:
                  contributor_value = str(contributor_ID)[4:-5]
                  contributor_name = str(contributor.find("username"))[10:-11]

                  # same usernames have commas, so we need to escape them
                  contributor_name = '"'+contributor_name+'"'
              else:
                  contributor_IP = contributor.find("ip")
                  contributor_value = str(contributor_IP)[4:-5]
                  contributor_name = "N/A"

          outFile.
↪write(f"{ID_value},{title_value},{rev_ID_value},{rev_time_value},{contributor_value},{contr
        nb_page_processed+=1
        # we show how much progress has been made
        if nb_page_processed % 100 == 0:
            progress_message(f"Processed {nb_page_processed} pages")

progress_message(f"Starting. Number of pages to process: {len(files)}")

for current_file in files:

    progress_message(f"Processing file {current_file}")

    extract_data(folder_in+current_file,folder_out)

progress_message("Done.")
```