PageSageXpert: A Wikipedia-based Application

Rhema K. Marneni
Rutgers University
Piscataway, NJ, USA
rkm110@scarletmail.rutgers.edu

Bhargavi Chinthapatla
Rutgers University
Piscataway, NJ, USA
bc837@scarletmail.rutgers.edu

Saman Ebrahimi
Rutgers University
Piscataway, NJ, USA
se308@scarletmail.rutgers.edu

***Abstract*— We developed an innovative application designed to analyze and enhance the accessibility of Wikipedia data. Leveraging Apache Spark and employing the Block Stripe Analysis technique for PageRank computation, we processed the entirety of Wikipedia content up to December 1, 2019. The optimization of the computation process across 1000 partitions ensures a robust and scalable processing pipeline, effectively handling the massive volume of Wikipedia data. This project not only showcases the successful implementation of a powerful search engine and data visualization, but also highlights the efficiency gains achieved through the strategic use of Apache Spark and the Block Stripe Analysis technique. The outcomes of this work contribute to the field of information retrieval and processing large-scale datasets, providing a valuable tool for users seeking structured access to Wikipedia knowledge.**

***Keywords*— Apache Spark, BART, Block Stripe Algorithm, Django framework, Graph building, Graph City, information retrieval, Massive Data, PageRank, Pyspark, Search Engine, Summarization, Wikidata, Wikipedia**

I. Project Description

In this project, we recognize Wikipedia as an unparalleled and expansive source of information, offering users an extensive array of topics to explore. However, the sheer volume of data can overwhelm users, leading them to get lost in the vast sea of information. Although Wikipedia provides a search engine, it primarily relies on keyword matching, directing users to the most relevant page based on their keyword input. Acknowledging the enormity of Wikipedia's database, comprising millions of pages, our project aims to harness the full potential of this massive dataset.

To address the challenges posed by the scale and complexity of Wikipedia data, we proposed the development of a search engine. This engine retrieves relevant articles based on user queries, presenting them in descending order of pagerank value to prioritize the most influential and pertinent content. Furthermore, we employed natural language processing (NLP) techniques to generate concise summaries for the selected articles, enhancing user accessibility and comprehension. Additionally, we aimed to incorporate data visualization techniques to provide users with a convenient and intuitive understanding of the information landscape within Wikipedia. Through these endeavors, our project seeks to streamline the exploration of Wikipedia data, making it more accessible, informative, and user-friendly.

Target Users: Any general public. We expect journalists and researchers to particularly benefit greatly from this application as it aids in efficient navigation through wikipedia data.

II. Data Collection and Preprocessing

The Kensho Derived Wikimedia Dataset (KDWD)[1] serves as a refined resource, condensing Wikipedia and Wikidata data for optimal use in natural language processing (NLP) research. Released under the CC BY-SA 3.0 license, this dataset combines a link-annotated English Wikipedia corpus and a compact sample of the Wikidata knowledge base. Versioned by raw Wikimedia snapshot dates, the current release, "kdwd_enwiki_20191201_wikidata_20191202," reflects its construction from snapshots of December 1 and 2, 2019. With meticulous curation, the KDWD creates three interconnected layers: a plain text Wikipedia corpus, link annotations, and connections to Wikidata items. This dataset proves indispensable for exploring the intricate relationships between Wikipedia content and Wikidata entities[1], catering to the specific demands of NLP applications. The KDWD's structured format ensures efficient parsing, providing a valuable bridge between large-scale knowledge bases and advanced NLP research.

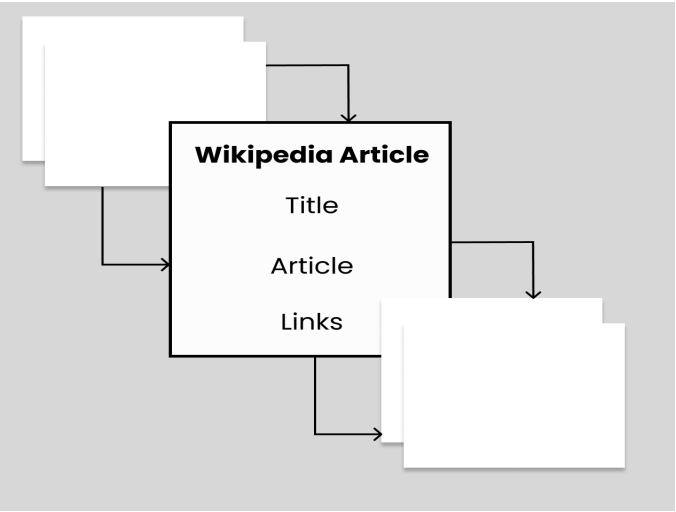

Fig. 1. Illustration of Pages on Wikipedia Platform

Data preprocessing:

- **Data Selection:** Select Wikipedia pages from the (Main/Article) namespace, excluding disambiguation pages.Include only pages with an associated Wikidata

item and exclude Wikimedia internal items.

- **Wikipedia Corpus Construction:** Construct a corpus of link-annotated text by parsing English Wikipedia pages meeting the selection criteria. Structure the data in a JSON Lines file, with each page represented as an object containing page metadata and a list of section objects.

- **Section Parsing:** Parse each section's wikitext markup into plaintext, creating a natural language representation. Identify text spans representing links using attributes like link offsets, link lengths, and target page IDs.

- **Data Validation and Quality Control:** Perform rigorous validation checks to ensure the integrity of the processed data. Address discrepancies and anomalies arising from the challenges in parsing Wikimedia source files and wikitext.

- Project Timeline and Division of Labor.

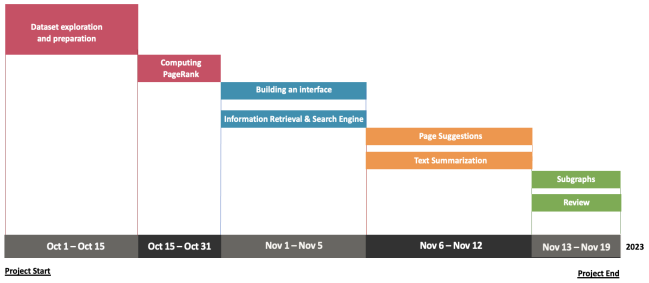

Fig. 2. 8 weeks Gantt Chart that was followed to complete the PageSageXpert project.

Bhargavi Chinthapatla: Computing Pagerank, subgraph of web pages, Information Retrieval and Search Engines, Suggestions, User Interface.
Rhema Marneni: Computing Pagerank, Graph city visualization, Information Retrieval and Search Engines, Text Summarisation, User Interface.
Saman Ebrahimi: Computing Pagerank, Text summarisation.

III. Algorithm Development

Figure 3 and figure 4 show the conceptual flow of the project and the ER diagram respectively. We perform pagerank computation, text summarization and display Wikipedia page suggestions that appear on the user's homepage upon login.

A. Block Stripe Technique for PageRank Computation

The PageRank algorithm, commonly employed for web page ranking, often utilizes power iteration for iterative computation.

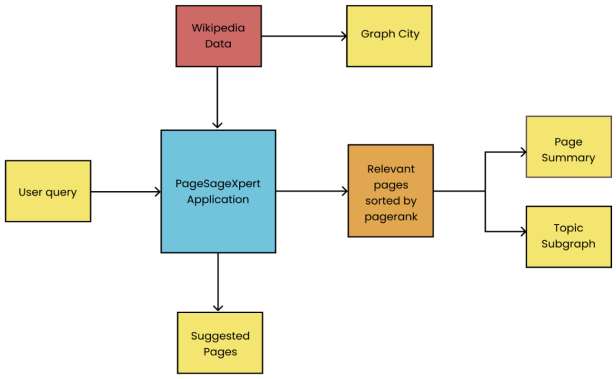

Fig. 3. Conceptual Flow of the Project Implementation

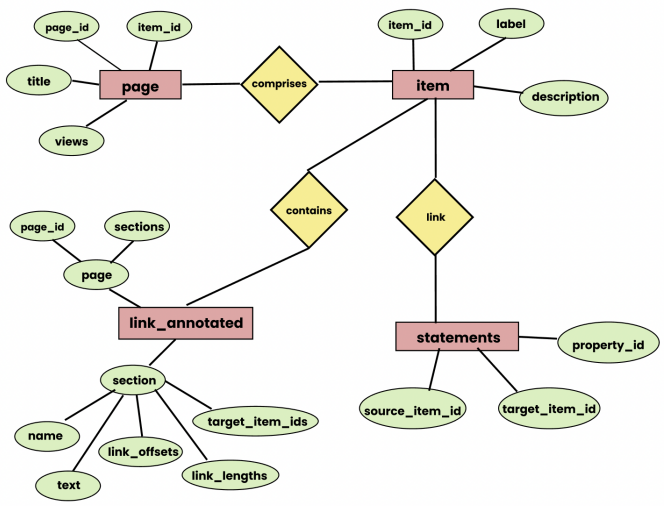

Fig. 4. ER Diagram

However, the straightforward approach of holding the entire stochastic web matrix A and the vectors r_old and r_new in memory for the computation of

$$r\_new = A \cdot r\_old \qquad (1)$$

becomes impractical as the matrix size increases. It demands a large number of scans and inefficient memory management. The Block Stripe technique emerges as a highly efficient alternative, addressing memory constraints even when the vector r_new may not fit into memory. This technique involves sparse matrix encoding, where space is roughly proportional to the number of links. r_new is divided into blocks and the web matrix is divided into stripes according to each r_new block (figure 6). It achieves this by creating an adjacency list representing the graph, incorporating crucial information about source nodes, degrees, and target node IDs. To improve the convergence of the PageRank algorithm, a damping factor (β) is employed that avoids spider traps and deadends.

Let β be the damping factor, M be a sparse matrix of size N ×

N, *r_old* be a vector of size N representing the PageRank values in the previous iteration, and *r_new* be a vector of size N representing the updated PageRank values in the current iteration. The algorithm is shown in figure 5.

Algorithm 1 PageRank Algorithm with Block Stripe Technique

Initialize r_{old} to $\frac{1}{\beta}$
Initialize all entries of r_{new} to $\frac{1-\beta}{N}$ and break into k blocks that fit in memory
Break M into stripes where each stripe contains only destination nodes in the corresponding block of r_{new}
for each stripe of M **do**
    for each page p (out-degree n) in that stripe **do**
        Read into memory: p, n, $dest_1, \ldots, dest_n$, $r_{old}(p)$
        for $j = 1$ to n **do**
            $r_{new}(dest_j)$ += $\beta \cdot \frac{r_{old}(p)}{n}$
        end for
    end for
end for

Fig. 5 Block Stripe Algorithm for PageRank

The Block Stripe technique thus presents an optimized and scalable solution for the challenges posed by the increasing size and complexity of web graphs.

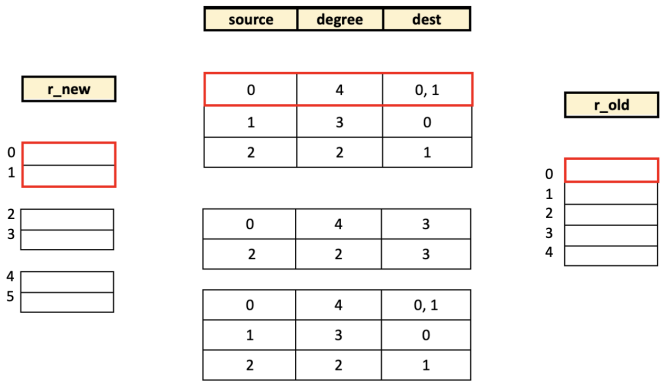

Fig. 6 Block Stripe Analysis for PageRank

```
[(330, 30, [5282, 5282, 2005352, 14851243, 7645794, 7513173, 6685836, 7484079, 7513173, 668583
6, 7484079, 7645794, 450022, 9337977, 4140629, 177181, 41881, 187603, 19344515, 58959, 1743562,
9235807, 2005352, 7513173, 6685836, 7484079, 7645794, 975073, 14851243, 23538754])]
```

Fig. 7. One page inside a stripe. The elements from left to right are: page_id, its outdegree and its destination pages.

For the wikipedia dataset, the adjacency list is computed and page information is obtained as shown in figure 7.

B. BART Summarization for each search result

For each of the search results displayed, we provide a short summary of the entire wikipedia page. This way, the user does not need to actually visit each wikipedia page and attempt to read the information in its entirety. We employed the BART model.

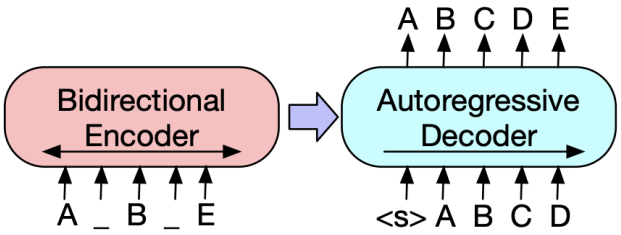

Fig. 8. BART Model

The BART model for summarization, short for "Bidirectional and Auto-Regressive Transformers," represents a significant advancement in natural language processing and text summarization. At its core, BART leverages the Transformer architecture, which has proven to be highly successful in various natural language processing tasks. The bidirectional aspect of BART refers to the model's ability to understand context from both directions—left-to-right and right-to-left. This bidirectional pretraining allows the model to capture rich contextual information and relationships within the text, enabling a more comprehensive understanding of the input data. By combining bidirectional pretraining with autoregressive training, BART achieves a balance between capturing global context and generating sequential outputs. The BART model has demonstrated good performance in our application.

For example, for the summary generated for the wikipedia page "Anarchism" is as follows:

"Anarchism is an anti-authoritarian political and social philosophy that rejects hierarchies deemed unjust and advocates their replacement with self-managed, self-governed societies based on voluntary, cooperative institutions. These institutions are often described as stateless societies, although several authors have defined them more specifically as distinct institutions based on non-hierarchical or free associations. Anarchism's central disagreement with other ideologies is that it holds the state to be undesirable, unnecessary, and harmful. Anarchism is usually placed on the far-left of the political spectrum, and much of its economics and legal philosophy reflect anti-communitarian interpretations of communism, collectivism, syndicalism, mutualism, or participatory economics. As anarchism does not offer a fixed body of doctrine from a single particular worldview, many anarchist types and traditions exist and varieties of anarchy diverge widely."

C. Suggested Pages for each user

Our application features an innovative algorithmic framework aimed at elevating user engagement through the dynamic suggestion of personalized Wikipedia articles. This is achieved by conducting real-time analyses on securely stored search

histories. The algorithm capitalizes on cosine similarity to assess the likeness between user search history vectors and an extensive array of pages. The process encompasses data preprocessing, cosine similarity calculation, page ranking, and the subsequent display of personalized recommendations upon user login. The algorithm is designed to scale seamlessly with growing user databases. Key attributes of the algorithm include robust security measures that prioritize user data privacy, along with the seamless integration of industry-standard protocols. By tailoring content to individual preferences, the algorithm significantly enhances the user experience, fostering a dynamic and engaging digital environment.

IV. Application Development

We have used the Django web framework implemented in this project, which seamlessly organizes the client-server architecture using the MVT (Model, View, Template) design pattern[7]. To enhance data retrieval speed, indexing is strategically applied, particularly for tables with substantial record volumes. This indexing strategy ensures expeditious and responsive access to the dataset.

Within the MVT pattern, Django's View is embodied in Python functions addressing client calls, while Templates, comprising HTML, CSS, and JavaScript, deliver visually appealing and interactive web pages. Leveraging the Django web framework, driven by the Python language, this project benefits from automatic connections and reframing, contributing to an efficient and professional web application. This integration aligns with industry best practices, providing a robust platform for data management and presentation.

Several essential libraries were used as listed below:
Pyspark
Django
Numpy
Pandas
Matplotlib
NltK

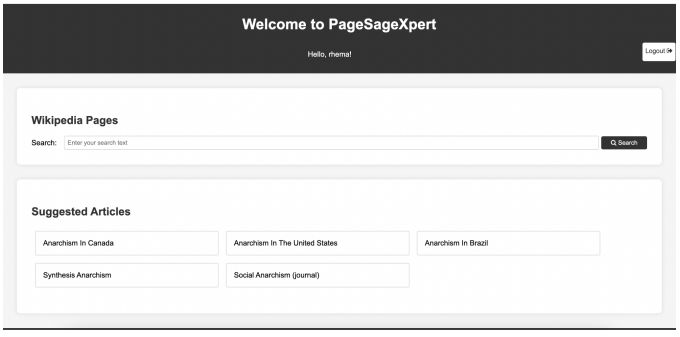

Fig. 9. PageSageXpert Homepage upon user login

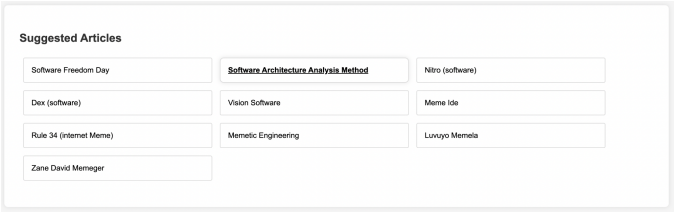

Fig. 10. Suggested Pages based on Recent Searches

Upon login, the user is presented with 10 suggested pages based on their recent search history. First time users have no recommendations until they begin their search.

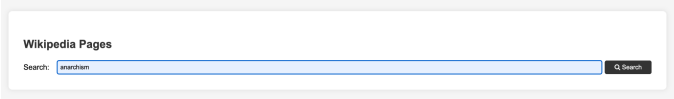

Fig. 11. Search functionality for Information Retrieval. Ex: "Anarchism"

The search functionality fetches the relevant search results based on PageRank.

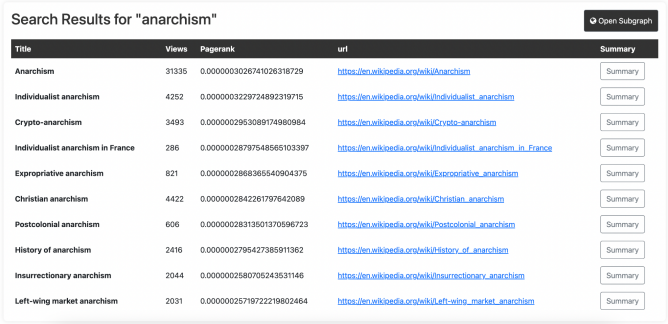

Fig. 12. Search Results for "Anarchism"

The top 10 results displayed are sorted in descending order of pageRank values. Users can choose which page's summary they would like to read. Other information like the number of views, and the URL to its corresponding wikipedia page is also provided.

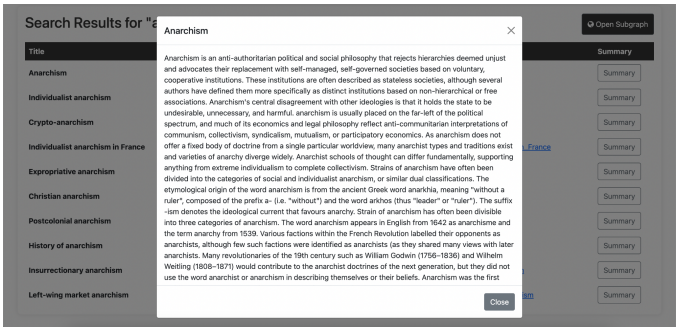

Fig. 13. Summary generated by the BART model for the page "Anarchism"

A summary of the entire wikipedia article is provided for each search result. Information is retrieved from each section

of the article, concatenated, tokenized and summarized for the user.

IV. Visualizations

A. Graph City of Wikidata

We transformed the vast wikidata information into a visual representation of a "Graph City." [2][3][4][5]. The graph city visualized 51,166,572 vertices and 134,637,197 edges, featuring a total of 56 distinct buildings and 39 unique peel values[6]. The tallest building in the Graph City had 258 floors and a peel value of 3. Notably, we observed that the taller buildings exhibited lower peel values and lower density, while the smaller buildings showcased higher peel values and higher density.

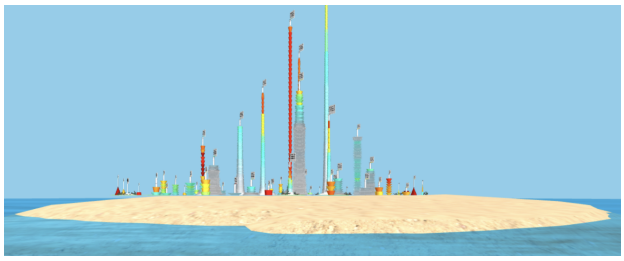

Fig. 14. Graph City Visualization of entire dataset

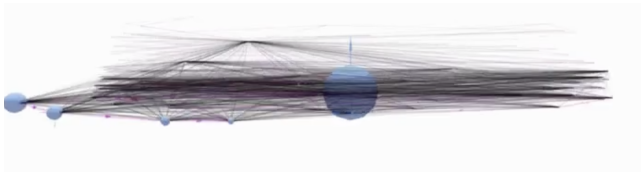

Fig. 15. Graph City Inner View of our tallest building

B. Subgraphs of Search Results

The application processes the user's search query to rank and present the top 10 most relevant pages based on their PageRank values in descending order. This prioritization ensures that users quickly access the most influential and contextually significant information. The resulting pages are dynamically represented within a 3D force-directed graph. This graph not only showcases the interconnectedness of each page within the search results but also reveals the relationships between these pages and their respective destination nodes. We use a vibrant color scheme, where each page's cluster is distinctly marked, facilitating a visual understanding of the conceptual groupings present in the results.

Notably, the application accounts for semantic similarities among the pages, resulting in multiple links between closely related content. In such instances, the subgraph becomes denser, offering a visual cue to the user about the richness and interconnected nature of the retrieved information. For instance, Figure 16 shows a subgraph derived from search results for the keyword "Anarchism."

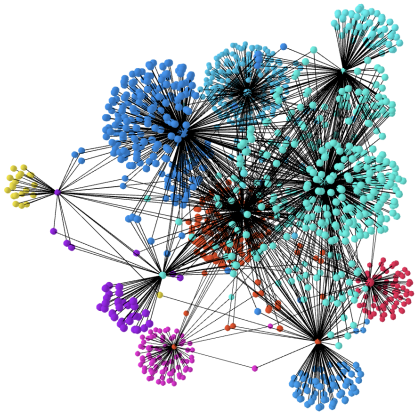

Fig. 16. Subgraph of "Anarchism" search results

V. Conclusion

In conclusion, our application has successfully met its objectives. The PageRank algorithm provided accurate results, and the search functionality operates as intended. The utilization of Apache Spark significantly improved processing efficiency, especially as it handled our large dataset. The incorporation of text summarization enhances user efficiency, allowing for quick article comprehension. Additionally, the suggested pages feature, based on recent search history, facilitates user navigation in a personalized manner. Overall, our application provides a convenient and efficient means for information retrieval, allowing users to skim through articles without the need for extensive reading. The visualizations have offered valuable insights, contributing to a clearer understanding of our dataset.

VI. Future Work

- Enhance the visualization component of subgraphs with interactive features, allowing users to explore and interact with the graph representation dynamically by navigating the page's Wikipedia URL.
- Apply centrality measures that allow users to gain insights on how connected the nodes (pages) are or how relevant the search results are to each other.
- Introduce a Deep Learning perspective by adding a chatbot to leverage the full potential of the dataset.
- The Wikimedia Dataset is rich in information. Further tools and processing can be applied to gain deeper insights.

VI. References

[1] Kensho Research. (2020). "Kensho Derived Wikimedia Data," distributed by Kaggle, https://www.kaggle.com/datasets/kenshoresearch/kensho-derived-wikimedia-data.

[2] Abello, J., Zhang, H., Nakhimovich, D., Han, C., & Aanjaneya, M. (2022). "Giga Graph Cities: Their Buckets, Buildings, Waves, and Fragments." IEEE Computer Graphics and Applications, 42, 53-64.

[3] Abello, J., Nakhimovich, D., Han, C., & Aanjaneya, M. (2021). "Graph Cities: Their Buildings, Waves, and Fragments." In EDBT/ICDT Workshops.

[4] Abello, J., & Nakhimovich, D. (2020). "Graph Waves." Big Data Res., 29, 100327.

[5] Abello, J., & Queyroi, F. (2013). "Fixed points of graph peeling." In 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), 256-263.

[6] "Graph City of PageSageXpert: A Wikipedia-based Application" [Video]. YouTube. Retrieved December 13, 2023, from https://youtu.be/azRk8F7-KRE.

[7] “PageSageXpert Project Demo” [Video]. YouTube. Retrieved December 13, 2023, from https://youtu.be/Z4BJXNqqF4M.