Final Essay

Course: Mining Massive Data Sets

Huynh Dang Khoa
Department of Computer Science
Ton Duc Thang University
Ho Chi Minh City
522h0104@student.tdtu.edu.vn

Chau Bao Nhan
Department of Computer Science
Ton Duc Thang University
Ho Chi Minh City
522h0093@student.tdtu.edu.vn

Nguyen Le Hai Long
Department of Computer Science
Ton Duc Thang University
Ho Chi Minh City
522h0125@student.tdtu.edu.vn

Nguyen Thanh An
Department of Computer Science
Ton Duc Thang University
Ho Chi Minh City
nguyenthanhan@tdtu.edu.vn

***Abstract*—This project encompasses four key tasks aimed at exploring large-scale data mining techniques. First, hierarchical clustering in non-Euclidean space was implemented using a custom agglomerative clustering class based on Jaccard distance and 4-shingle tokenization. Next, we conducted a gold price prediction task using PySpark's linear regression model, where the model was trained on historical gold price data. In the third task, the CUR matrix decomposition technique was applied to reduce feature dimensionality, followed by a comparative analysis of prediction accuracy before and after dimensionality reduction. The fourth task involved implementing Google's PageRank algorithm using PySpark to rank crawled web pages from the it.tdtu.edu.vn domain based on their importance. Finally, a comprehensive report was compiled in the IEEE conference template format. The results from all tasks were illustrated using appropriate visualizations such as bar charts and line graphs. All tasks were completed successfully and demonstrate practical applications of big data techniques using PySpark and clustering algorithms.**

***Index Terms*—Big data, Mining Massive Data Sets, Agglomerative Clustering, CUR algorithm, PageRank algorithm**

I. Introduction

In the era of big data, the ability to efficiently store, process, and analyze massive datasets has become increasingly vital. This project, conducted as part of the "Mining Massive Data Sets" course at Ton Duc Thang University, aims to explore and implement scalable data mining techniques using PySpark and custom algorithms. The project is structured into four main tasks, each addressing a fundamental challenge in data mining.

The first task focuses on hierarchical clustering in a non-Euclidean space, where we apply 4-shingle tokenization and Jaccard distance to group similar strings using a bottom-up agglomerative approach. The second task tackles time series prediction by building a linear regression model to forecast gold prices based on historical data. In the third task, the CUR matrix decomposition technique is used to reduce the dimensionality of features, enabling a comparative study of model performance before and after reduction.The fourth task implements the PageRank algorithm to analyze the structure of web pages within a given domain and determine their relative importance. Finally, all findings and results are summarized in a structured report using the IEEE conference paper format. Through these tasks, the project demonstrates practical applications of big data mining techniques and their effectiveness in solving real-world problems.

II. Task 1: Hierarchical clustering in non-Euclidean spaces

A. Objective

The objective of this task is to perform hierarchical clustering on textual data using a non-Euclidean similarity measure. Instead of relying on conventional Euclidean distance, we utilize the **Jaccard distance** computed on **4-shingles** extracted from character strings. This setup simulates clustering in a discrete, symbolic space where traditional geometric assumptions do not hold.

B. Synthetic Dataset Generation

To evaluate the algorithm, we generate a synthetic dataset consisting of 10,000 strings composed of lowercase alphabetic characters. The dataset generation follows these steps:

- A pool of 20 unique 4-character substrings (shingles) is created from random lowercase letters.
- A set of base strings (default: 20) is generated by concatenating shingles from the pool, with a randomly chosen length between 32 and 64 characters.
- Each sample string is produced by randomly selecting a base string and introducing character-level mutations at a fixed mutation rate (default: 3%). This simulates noisy, but semantically related, textual inputs.
- For each resulting string, we extract all contiguous substrings of length 4 (4-shingles) to represent it as a set.

Example: Original string: "abcdefghijk" Extracted shingles: "abcd", "bcde", "cdef", "defg", "efgh", "fghi", "ghij", "hijk"

C. Jaccard Distance

To compare two strings, we use the Jaccard distance between their corresponding sets of 4-shingles. Given two sets A and B, the Jaccard similarity is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Then, the Jaccard distance is:

$$D(A, B) = 1 - J(A, B)$$

This measure captures the dissimilarity between sets and is particularly effective for high-dimensional sparse data, such as tokenized strings.

D. Clustering Algorithm

We implement a bottom-up agglomerative clustering algorithm using Jaccard distance, suitable for non-Euclidean data. Each string initially forms its own cluster. At each step:

1) Compute all pairwise Jaccard distances and store them in a min-heap.
2) Iteratively extract the closest pair of clusters.
3) Merge them only if the resulting cluster's diameter (max internal distance) is below a predefined threshold.
4) Update the heap with new distances after merging.
5) Repeat until no further merges are allowed.

Clustroid: Within each cluster, the clustroid is the element minimizing the maximum distance to all other members, acting as its representative.

E. Visualization and Analysis

We report:

- Total number of clusters.
- Clustroid of each cluster.
- A bar chart of average intra-cluster distances to assess compactness.

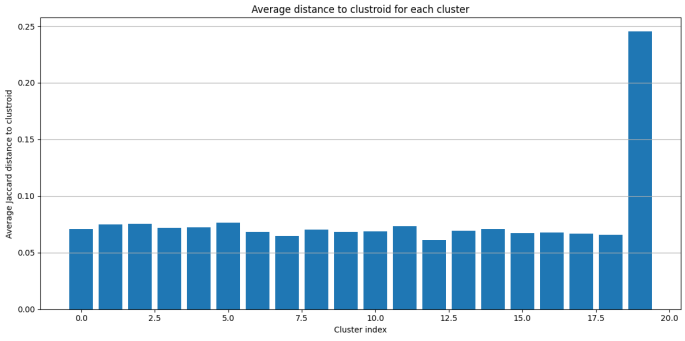

Fig. 1. Average distance from each point to clustroid from each clusters

III. Task 2: Linear Regression for Gold Price Prediction

A. Data Preprocessing and Feature Engineering

The dataset contains daily gold prices in Vietnam from August 1, 2009 to January 1, 2025, with three columns: `Date`, `Buy Price`, and `Sell Price`. The goal is to predict the gold price at day t based on the previous 10 days' prices.

We performed the following steps for data preprocessing and feature generation:

- The data was sorted chronologically by `Date` to maintain temporal order.
- For each day t, 10 lag features were created, representing the prices of the previous 10 days. For instance, `Buy_1` is the buy price at day $t-1$, and `Buy_10` is the buy price at day $t-10$.
- Rows containing null values in any lag features (arising due to missing prior days at the start) were removed to ensure complete feature vectors.
- This lag feature generation was performed separately for `Buy Price` and `Sell Price`, producing two datasets for buy and sell price prediction.
- All lag features were cast to `DoubleType` to fit the input type requirement of the PySpark ML models.
- We combined the lag features into a single `features` vector using PySpark's `VectorAssembler`.
- The target variable `label` was set as the buy price or sell price at day t, respectively, cast to double.
- Finally, each dataset contained the columns `Date`, `features`, and `label`.

This procedure transformed the time series data into a supervised learning format suitable for linear regression.

B. Data Splitting

Each dataset was randomly split into training and testing sets with a ratio of 70:30, ensuring that the model's performance can be evaluated on unseen data.

After training, the model's performance was evaluated on both training and test sets using Root Mean Squared Error (RMSE).

C. Results Visualization

The model was trained on the training set, taking the 10-day lag feature vector as input and predicting the price of day t. During training, we recorded the loss metrics for each iteration to monitor convergence. The plots below show the predicted prices for both buy and sell targets.

From the figures, we can observe the overall trend captured by the model, demonstrating its ability to follow fluctuations in the gold price to a reasonable extent. Although the predictions may not be perfectly aligned with the actual price curve at every single point, the model is able to capture both short-term variations and long-term movements.

Specifically, in the prediction for buy price, the model tends to slightly underpredict during sharp uptrends and overpredict during rapid declines, which is common in time-series forecasting due to lagged input features. On the other hand, the prediction for sell price shows a relatively smoother pattern, indicating that the model might be generalizing better in this case.

These visualizations serve as a crucial diagnostic tool, helping us to assess whether the model is underfitting or overfitting, and whether it is responsive enough to capture volatility in the

market. Further improvements can be made by experimenting with deeper models, regularization, or incorporating external variables such as currency exchange rates or global gold market trends.

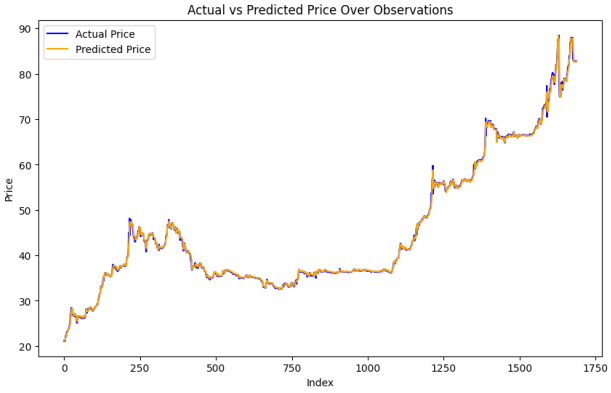

Fig. 2. Prediction for buy price

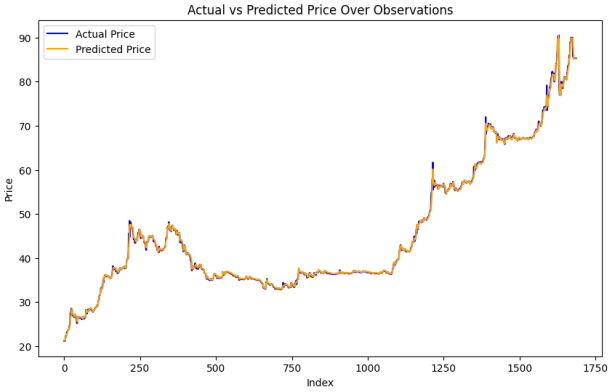

Fig. 3. Prediction for sell price

IV. Task 3: CUR - Dimensionality Reduction

A. Data Preprocessing

The dataset used for Task 3 is identical to that in Task 2, containing daily gold price in Vietnam from August 1, 2009 to January 1, 2025, with columns.

To prepare the data for CUR decomposition, the feature vectors(10-dimensional) from both Buy Price and Sell Price data are converted from `PySpark ML`'s vector format `PySpark MLLIB`'s vector format as required by the `RowMatrix API` used in the CUR algorithm. I will show table of buy price before CUR and table of Sell price before CUR:

Date	features (10D)	label
2009-08-11	[21.13, 21.13, 21.13, ..., 21.13, 21.13, 21.13]	21.13
2009-08-12	[21.13, 21.13, 21.13, ..., 21.13, 21.13, 21.13]	21.13
...	...	...

TABLE I
Table of Buy Price before CUR

Date	features (10D)	label
2009-08-11	[21.19, 21.19, 21.19, ..., 21.19, 21.19, 21.19]	21.19
2009-08-11	[21.19, 21.19, 21.19, ..., 21.19, 21.19, 21.19]	21.19
...	...	...

TABLE II
Table of Sell Price before CUR

B. CUR-based Feature Reduction

We implemented a custom CUR-based dimensionality reduction algorithm to reduce the feature dimension from 10 to 5.

Instead of reconstructing the original matrix $A \approx CUR$, we apply CUR decomposition to generate low-dimensional **row embeddings** of shape $(m \times 5)$ using the following strategy:

- **Select** C: Select 5 columns from A with the highest squared column norms using function `select_columns`:

$$\text{Importance}(j) = \sum_{i=1}^{m} (A_{ij})^2$$

  Columns are sampled with probabilities proportional to their squared norms.
- **Select** R: Select 5 rows from A with the highest squared row norms using function `select_rows`:

$$\text{Importance}(i) = \sum_{j=1}^{n} (A_{ij})^2$$

- **Compute** U: We define the intersection matrix $W = C^T R$, and apply SVD:

$$W = U_W \Sigma_W V_W^T$$

  Then compute:

$$\Sigma_W^+ = \text{diag}\left(\begin{cases} 1/\sigma_i & \text{if } \sigma_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \right)$$

  The core transformation matrix is computed as:

$$U = V_W \cdot (\Sigma_W^+)^2 \cdot V_W^T$$

- **Row embeddings:** Finally, we project the original rows onto the reduced space:

$$A_{\text{embed}} = C \cdot U$$

  Each row in A_{embed} is a 5-dimensional embedding corresponding to the original input row.

The new 5-dimensional data is generated as $RowEmbedding = C \cdot U$ using the `transform` function.

C. Model Training and Evaluation

We used PySpark's built-in `LinearRegression` model from the `pyspark.ml.regression` module to train predictive models for both buy and sell prices.

Date	features (5D)	label
2009-08-11	[3.07776930E-4, ..., 90994.11871588]	21.13
2009-08-12	[3.07776930E-4, ..., 90994.11871588]	21.13
...	...	...

TABLE III
TABLE OF BUY PRICE AFTER CUR

Date	features (5D)	label
2009-08-11	[2.98625222E-4, ..., 93086.35967722]	21.19
2009-08-12	[2.98625222E-4, ..., 93086.35967722]	21.19
...	...	...

TABLE IV
TABLE OF SELL PRICE AFTER CUR

D. Data Visualization

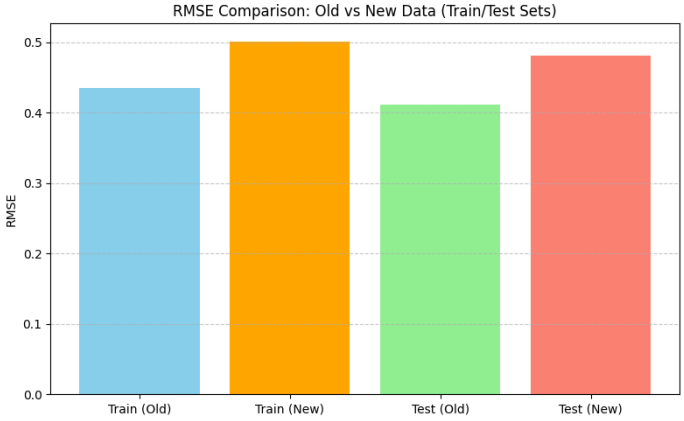

Fig. 4. Prediction performance for Buy Price

Buy Price RMSE:

- Training set: RMSE (Original): 0.434, RMSE (Reduced): 0.501
- Test set: RMSE (Original): 0.411, RMSE (Reduced): 0.480

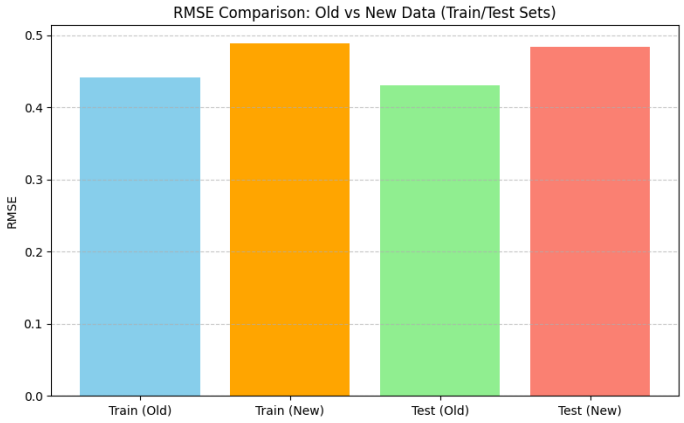

Fig. 5. Prediction performance for Sell Price

Sell Price RMSE:

- Training set: RMSE (Original): 0.440, RMSE (Reduced): 0.489
- Test set: RMSE (Original): 0.430, RMSE (Reduced): 0.483

E. Conclusion

Linear regression uses parameters: `maxIter=50`, `regParam=0.1`, `elasticNetParam=0.8`, tol=1×10^{-6}. Evaluation is done via RMSE.
CUR Decomposition reduces computational complexity but slightly RMSE (0.4-0.45 to 0.45-0.50). Alternative dimensionality reduction could be explored to improve accuracy

V. TASK 4: PAGERANKING – THE GOOGLE ALGORITHM

A. Introduction

Task 4 implements the PageRank algorithm to rank web from https://it.tdtu.edu.vn based on the link structure. A web crawler collected source-destination URL pairs, forming a directed graph stored in a PySpark DataFrame. The process involes crawling sub-pages, preprocessing the data to remove self-loops and duplicates and using PySpark to implement the PageRank Google algorigthm for page ranking. This task covers data collection, preporcessing, methodology and results.

B. Data Preprocessing

The dataset was crawled from https://it.tdtu.edu.vn using a Python-based crawler. It collected source-destination URL pairs, while filtering out static resources such as images, videos, stylesheets (CSS), JavaScript files, and other non-HTML content. The collected data was stored in a PySpark DataFrame with the schema:

```
(source_url: String, dest_url: String)
```

This structure forms a directed graph, where each edge represents a hyperlink from one page to another. After preprocessing and filtering, the resulting graph consists of approximately **39,831 edges** and **2,410 unique web pages (nodes)**.

To ensure consistency, URLs were normalized by:

- Removing URL fragments (e.g., `#section`)
- Converting all characters to lowercase
- Ensuring trailing slashes were consistent

A sample of the preprocessed data is shown in Table **??**. caption

Source URL	Destination URL
https://it.tdtu.edu.vn	https://it.tdtu.edu.vn/tin-tuc
https://it.tdtu.edu.vn	https://it.tdtu.edu.vn/sinh-vien
...	...

TABLE V
TABLE SHOW SOURCE URL AND DESTINATION URL

C. PageRank Algorithm

We stored these URL pairs as an edge list in a PySpark DataFrame, which represents a directed graph. The PageRank algorithm was then implemented as a PySpark class using the power iteration method, following the standard pseudo-code:

1) Initialize PageRank of each page to $1/N$ where N is the number of nodes.
2) At each iteration, update the PageRank of a node using:

$$PR(v) = \alpha \sum_{u \to v} \frac{PR(u)}{L(u)} + (1 - \alpha)\frac{1}{N}$$

3) The `pagerank` function implements the power iteration method with parameters: the `alpha=0.85`, `max_iter=100` and `tol=1.0e-6`.
4) Repeat until convergence or max iterations.

D. Results

The crawler processed **9230 pages** in approximately **17413.85 seconds** after **10 iterations**. The PageRank algorithm, applied to preprocessed data, ranked the top 10 URLs (illustrative scores):

URL	PageRank Score
https://it.tdtu.edu.vn/	0.037938
https://it.tdtu.edu.vn/en	0.036032
https://it.tdtu.edu.vn/sinh-vien	0.031238
https://it.tdtu.edu.vn/vien-chuc	0.031238
https://it.tdtu.edu.vn/tin-tuc/tuyen-dung	0.030735
https://it.tdtu.edu.vn/gioi-thieu	0.030671
https://it.tdtu.edu.vn/tin-tuc	0.030603
https://it.tdtu.edu.vn/giao-duc	0.030547
https://it.tdtu.edu.vn/tin-tuc-khoa	0.030429
https://it.tdtu.edu.vn/khoa-hoc-cong-nghe	0.030168

TABLE VI
PAGERANK SCORES OF TOP 10 PAGES

The homepage and key sections ranked highest, reflecting their central roles in the link structure

CONTRIBUTIONS

- Huynh Dang Khoa: 100%
- Chau Bao Nhan: 100%
- Nguyen Le Hai Long: 100%

SELF-EVALUATION

Completion of each task

- Task 1: 100%
- Task 2: 100%
- Task 3: 100%
- Task 4: 100%
- Task 5: 100%

CONCLUSION

In this project, we explored various essential techniques in mining massive data sets through four comprehensive tasks.

Task 1 focused on hierarchical clustering in non-Euclidean spaces. We successfully implemented an agglomerative clustering algorithm using the 4-shingle technique and Jaccard distance for string similarity. The clustering results were visualized using average distances to clustroids.

Task 2 involved predicting gold prices using linear regression in PySpark. We transformed historical gold price data into feature-label samples and trained a regression model, achieving good performance on both training and test sets. Training loss was tracked and illustrated via line charts.

Task 3 applied the CUR matrix decomposition technique for dimensionality reduction. Feature vectors from Task 2 were reduced from 10 to 5 dimensions, and the impact on regression performance was analyzed. We observed the trade-off between dimensionality and accuracy via comparative bar charts.

Task 4 implemented the PageRank algorithm using PySpark. We crawled the domain https://it.tdtu.edu.vn, constructed a graph of page links, and computed the importance of web pages. The resulting ranking reflects the centrality of each page within the domain.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to MSc.Nguyen Thanh An for his thorough guidance and support, which enabled our group to complete five task of this report.

REFERENCES

[1] Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). *Introduction to data mining* (2nd ed.). Pearson.

[2] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2020). *Mining of massive datasets* (3rd ed.). Cambridge University Press.

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

[4] Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4th ed.). Johns Hopkins University Press.

[5] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391–407.

[6] Dong, X. L., & Srivastava, D. (2020). Big data integration. *Synthesis Lectures on Data Management, 12*(1), 1–198. https://doi.org/10.2200/S01078ED1V01Y202001DTM061