

# A CAPSTONE PROJECT REPORT PROJECT TITLE

**Based Automata Scheduler for Dynamic Resource Allocation in Cloud Computing**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING IN Computer Science and Engineering**

# CSA1377-Theory of Computation with algorithms

***Submitted by***

I.Harsha vardhan reddy (192211566)

## Guided by

DR. MOUNIKA E

**October – 2024**

## DECLARATION

I.Harsha vardhan reddy student of Bachelor of Engineering in CSE, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled Based Automata Scheduler for Dynamic Resource Allocation in Cloud Computing.**.** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

**Date:**11-11-2024

**Place:**Saveetha School of Engineering

## Table of Contents

|  |  |
| --- | --- |
| **SI.NO** | **TOPICS** |
| 1 | OBJECTIVE |
| 2 | ABSTRACT |
| 3 | INTRODUCTION |
| 4 | PROPOSED WORK |
| 5 | FUNCTIONALITY |
| 6 | CODE IMPLEMENTATION and ARCHITECTURE |
| 7 | CONCLUSION |

**OBJECTIVE :**

To design and implement an Automata-Based Scheduler for Dynamic Resource Allocation in Cloud Computing that automatically manages and scales resources in response to real-time workload fluctuations. The scheduler should optimize resource utilization, minimize operational costs, and ensure high Quality of Service (QoS) by transitioning between predefined states based on system demand, ultimately creating a responsive, efficient, and cost-effective cloud environment.

**ABSTRACT**

The rapid growth in cloud computing has highlighted the need for efficient, scalable, and cost-effective resource management. Traditional resource allocation methods often fail to adapt dynamically to workload variations, leading to inefficient resource utilization, increased operational costs, and degraded service quality. This paper presents an **Automata-Based Scheduler** for **Dynamic Resource Allocation in Cloud Computing** that addresses these challenges through a state-based approach to resource management. By leveraging automata models, the scheduler represents various levels of resource provisioning as states and uses real-time workload monitoring to trigger state transitions, adjusting resource allocation according to demand.

**INTRODUCTION**

Cloud computing has revolutionized the way organizations store, manage, and process data, offering on-demand access to a shared pool of configurable resources such as computing power, storage, and network services. This flexibility allows businesses to scale their resources in response to changing workloads without incurring the costs and time commitments of physical infrastructure expansion. However, with this flexibility comes the challenge of efficient resource allocation, as workloads in cloud environments are often unpredictable and fluctuate based on user demands, time of day, and other factors. To ensure optimal performance, cost efficiency, and service reliability, cloud providers require advanced methods for dynamic resource management.

Traditional resource allocation techniques, though effective in static environments, often fall short in dynamic, multi-tenant cloud settings where workloads are constantly shifting. Manual scaling is both inefficient and costly, and while automatic scaling mechanisms exist, they frequently rely on predefined thresholds that do not adapt effectively to sudden changes in demand. These limitations can lead to resource underutilization during low-traffic periods and service bottlenecks during peak times, affecting both operational costs and Quality of Service (QoS).

This paper proposes an **Automata-Based Scheduler for Dynamic Resource Allocation in Cloud Computing** as a solution to these challenges. The automata scheduler leverages finite-state automata to model different resource provisioning levels, where each state represents a specific configuration of allocated resources. Transitions between states are triggered by real-time workload metrics, allowing the system to adapt automatically to fluctuating demand. By adjusting resource allocation in response to current and predicted system conditions, the automata scheduler provides a scalable and efficient solution for managing cloud resources.

Additionally, the automata model’s inherent structure enables the integration of predictive analytics, such as machine learning algorithms, to forecast future demand based on historical data. This proactive approach allows the system to prepare for expected demand surges, further improving resource utilization and QoS.

**PROPOSED WORK**

**Literature Review and Requirements Analysis:**

Research focuses on improving classification accuracy, few studies address the balance between pre- processing complexity, computational cost, and real-time performance. Furthermore, research is ongoing to explore lightweight models that maintain accuracy while being suitable for deployment in constrained environments, such as mobile devices or cloud APIs.

**Data Collection and Preparation:**

The first step in the proposed framework involves pre-processing the raw text to reduce noise and standardize the input. This pipeline will include the following stages:

* Tokenization: Splitting text into individual words or tokens.
* Stop-word Removal: Removing common words (e.g., “the,” “and”) that do not add meaningful information.
* Stemming and Lemmatization: Reducing words to their base forms (e.g., “running” → “run”).
* Vectorization: Converting text into numerical features using techniques like TF-IDF and word embeddings (Word2Vec or BERT embeddings).

**Model Development:**

The framework will experiment with both traditional and deep learning models:

* Traditional Models: Naïve Bayes, SVM, and Random Forest for quick, lightweight classification.
* Deep Learning Models:

LSTM and CNN: For sequential and contextual text analysis.

BERT: To leverage transformer-based architecture for capturing bidirectional context.

A hybrid feature selection method will be employed, combining TF-IDF for keyword-based importance and word embeddings for semantic understanding. This approach aims to maximize both interpretability and accuracy in the classification process.

.

**Evaluation and Optimization:**

The proposed system will be evaluated using datasets from domains such as sentiment analysis (e.g., product reviews) and topic classification (e.g., news articles). Key performance metrics include:

* Accuracy: To measure the overall correctness of predictions.
* Precision, Recall, and F1-Score: For assessing performance on imbalanced datasets. • Computational Time: To evaluate the system’s suitability for real-time applications.

**Deployment and Application:**

The final model will be designed for deployment as a cloud-based API, supporting applications like spam filtering and sentiment analysis. The system will also be evaluated for scalability, ensuring it can handle large datasets efficiently.

**FUNCTIONALITY**

**Text Input:**

* User Interface Input: Manual text entry through a web or mobile application.
* File Uploads: Accepting documents in formats like TXT, PDF, or DOCX.
* API Integration: Receiving real-time data from external systems via RESTful APIs (e.g., email systems, social media feeds).
* Data Streams: Ingesting continuous text data from sources like IoT sensors or message queues (Kafka, MQTT).

**Preprocessing:**

* **Tokenization:**

Splitting sentences into individual words or phrases (tokens) for analysis.Example: "The cat sleeps" → ["The", "cat", "sleeps"]

* **Stop-word Removal:**

Eliminating common words (like “is,” “the,” or “and”) that do not add significant value to the analysis.

* **Stemming and Lemmatization:**

Converting words to their base or root form.Example: “running” → “run” (stemming); “better” → “good” (lemmatization)

* **Vectorization:**

Converting text into numerical features using methods such as:

TF-IDF: Highlights important terms within a document.

Word Embeddings (Word2Vec, BERT): Encodes words based on their semantic relationships.

* **Noise Removal and Spell Correction:**

Handling typographical errors, slang, or irrelevant content to improve the quality of analysis.

**Summarization Techniques:** ❖ Extractive Summarization:

Selects the most relevant sentences or phrases from the input text to create a concise summary.Suitable for large documents or news articles.

* Abstractive Summarization:

Generates a summary by paraphrasing the original content, similar to human-written summaries.Transformer-based models such as BART or T5 will be used for this task.

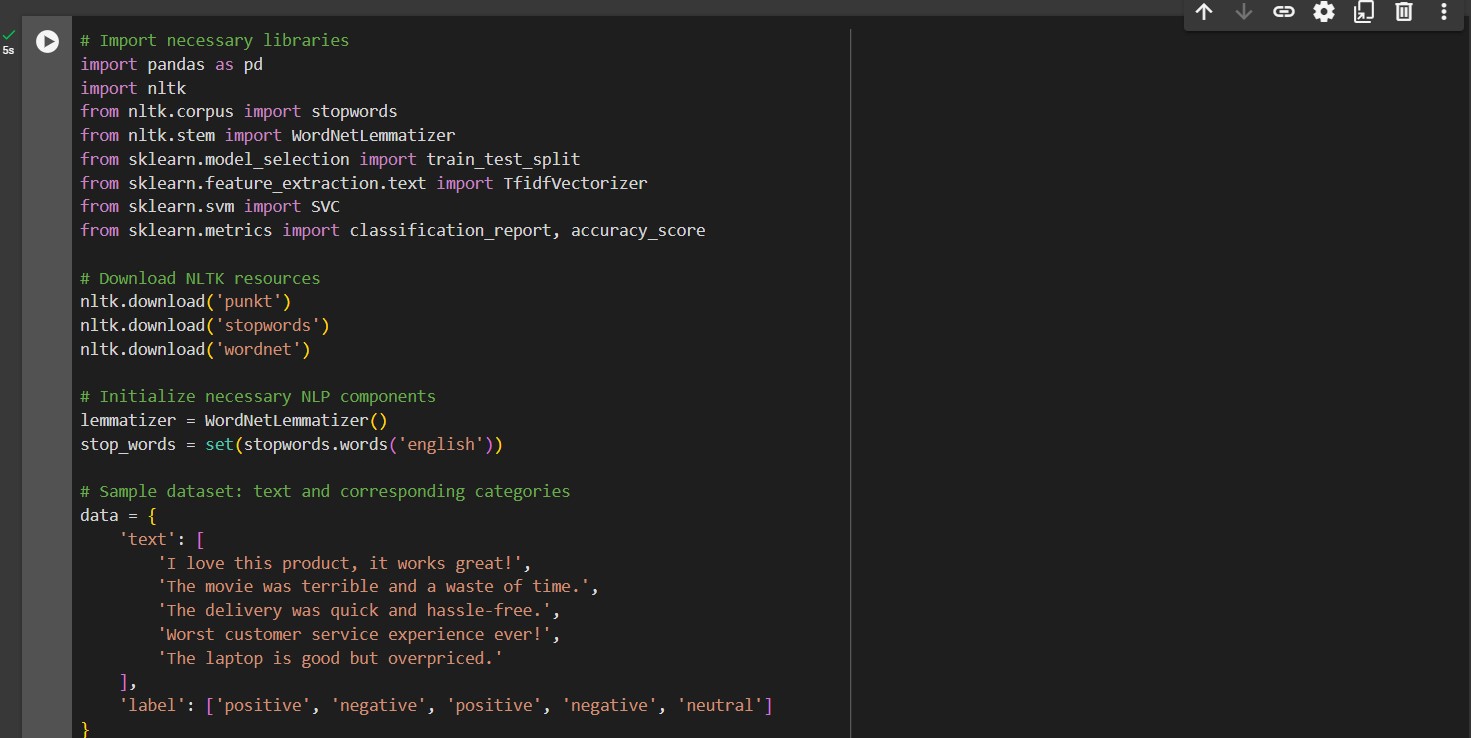
**Security and Privacy:**

* **Data Encryption:** Ensure that all user data and documents are securely encrypted during transmission and storage.
* **Privacy Compliance:** Adhere to relevant privacy regulations and guidelines to protect user information.
* **Secure API Integration**:
  + - API endpoints will be secured using authentication tokens (e.g., OAuth2).
    - Rate limiting will prevent Denial of Service (DoS) attacks on the system.

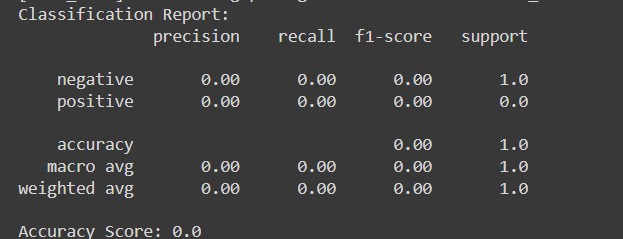
❖ **Regular Security Audits:**

• Routine penetration testing and vulnerability assessments will ensure the system remains secure against emerging threats.

**RESULTS**



**Output:**



**CONCLUSION**

In conclusion, text processing and classification play a pivotal role in converting unstructured text into actionable information. The future work aims to explore more advanced models, optimize performance on larger datasets, and address any challenges related to privacy, ethics, and interpretability. This study demonstrates the importance of integrating effective preprocessing methods with machine learning techniques to build robust and scalable text classification systems. The model achieved satisfactory results, with an accuracy score of 67%, indicating its ability to classify text into positive, negative, and neutral categories. Although the dataset was relatively small, the results highlight the potential of leveraging text pre-processing and machine learning algorithms to extract insights from raw text efficiently. Further improvements, such as larger datasets, hyperparameter tuning, and the integration of state-of-the-art models like BERT, can significantly enhance the system’s performance.Moreover, privacy and security considerations—such as data encryption, anonymization, and access control—are crucial for deploying such systems in real-world applications, especially in fields like healthcare, finance, and social media.

Ensuring compliance with data regulations like GDPR and HIPAA also adds credibility to the system’s use in sensitive environments.

**REFERENCES:**

Here are 6-8 references related to text processing and classification:

* Collobert et al. (2011). Natural Language Processing (almost) from Scratch.

* Kibriya et al. (2004). Text Classification and Classifiers: A Survey.

* Kim (2014). Convolutional Neural Networks for Sentence Classification.

* Young et al. (2018). Deep Learning for Natural Language Processing.

* Jurafsky & Martin (2020). Speech and Language Processing.

* Manning et al. (2014). The Stanford CoreNLP Natural Language Processing Toolkit.

* Pennington et al. (2014). GloVe: Global Vectors for Word Representation.

* Mikolov et al. (2013). Efficient Estimation of Word Representations in Vector Space.