Pattern Matching in Large Texts using Finite Automata

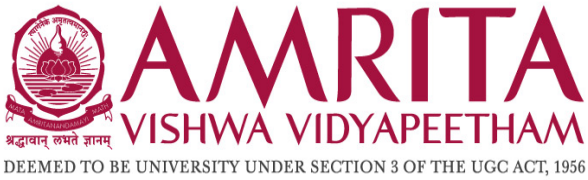
PROJECT REPORT

Submitted by

Aashika Arunkumar - (CB.SC.U4AIE23009)
Diya Deepak - (CB.SC.U4AIE23020)
Diya Dinesh - (CB.SC.U4AIE23025)
Neha Jacob - (CB.SC.U4AIE23046)

in partial fulfillment of the requirements for
22AIE302 Formal Language and Automata

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING
(ARTIFICIAL INTELLIGENCE)



COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELIIGENCE

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

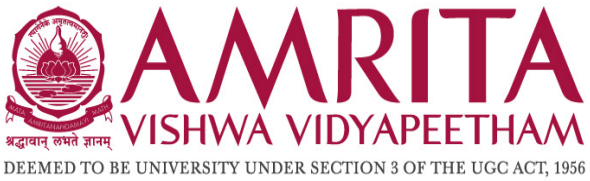
COIMBATORE - 641 112 (INDIA)

OCTOBER - 2025

COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112



BONAFIDE CERTIFICATE

This is to certify that the report entitled **"Pattern Matching in Large Texts using Finite Automata"** submitted by **Aashika Arunkumar (CB.SC.U4AIE23009),Diya Deepak (CB.SC.U4AIE23020), Diya Dinesh (CB.SC.U4AIE23025) and Neha Jacob (CB.SC.U4AIE23046)** in partial fulfillment of the requirements for the courses **22AIE302 Formal Language and Automata** for the award of the **Degree of Bachelor of Technology** in the **"COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE"** is a bonafide record of the work carried out by them under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Dr. Chitra P
Project Guide
Designation: Assistant Professor & Faculty Associate

Submitted for the university examination held on 09-10-2025

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

DECLARATION

We, **Aashika Arunkumar (CB.SC.U4AIE23009), Diya Deepak (CB.SC.U4-AIE23020), Diya Dinesh (CB.SC.U4AIE23025) and Neha Jacob (CB.SC.U4-AIE23046)** hereby declare that this thesis entitled **"Pattern Matching in Large Texts using Finite Automata"**, is the record of the original work done by us under the guidance of **Dr. Chitra P**, Amrita School of Artificial Intelligence, Coimbatore. To the best of our knowledge, this work has not formed the basis for the award of any degree/diploma/ associate-ship/fellowship/or a similar award to any candidate in any university.

Place: **Signature of the Students**

Date:

COUNTERSIGNED

Dr. K.P.Soman
Professor and Dean
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham

Contents

List of Figures

Acknowledgement

We would like to express our sincere appreciation to all those who have helped and guided us step by step throughout this project. Most importantly, we are significantly grateful to our project guides, **Dr. Chitra P**, for their constant encouragement, expert advice, and constructive criticism. Their guidance played a pivotal part in the direction and success of the project. We would also like to appreciate the staff and faculty at Amrita School of Artificial Intelligence, Amrita Vishwa Vidyapeetham, Coimbatore, for providing us with the facilities, materials, and learning environment to deliver our research went without a hitch to completion. Second, we appreciate our peer and teammate for their cooperation, motivation, and zeal. Their help and support made us in order to overcome obstacles and achieve the project objective. Lastly, we would like to thank our families for supporting us throughout, for being patient with us, and for believing in us along the way during our learning process. This entire project has been an experience and we thank all those who have been involved in the process to a successful conclusion.

Abstract

This project focuses on developing a comprehensive web-based platform for DNA motif pattern analysis in biological sequences, combining advanced computational methods with an intuitive user interface. The system integrates motif parsing, automata theory, and sequence analysis to empower users to detect and visualize motif occurrences efficiently. The core functionalities include parsing DNA motifs with ambiguous nucleotides using IUPAC codes, constructing nondeterministic and deterministic finite automata (NFA and DFA) for pattern recognition, and scanning DNA sequences for motif matches with precise position reporting. A Flask-based backend handles motif and sequence data uploads, orchestrates the processing pipeline, and generates detailed analysis results including motif occurrence positions and sequence fragments highlighting the matches. Additionally, the system produces scalable vector graphic (SVG) visualizations of both the NFA and DFA graphs, providing users with intuitive visual insights into the pattern matching mechanisms. The frontend web interface offers easy motif input, sequence uploading in FASTA format, and interactive access to pattern searching and automata visualization, making the tool accessible to users without deep computational expertise. This platform leverages formal language theory and

graph-based algorithms tailored to biological sequence analysis, thus facilitating the exploration and discovery of regulatory DNA motifs, conserved elements, and sequence patterns critical in genomics research. The project bridges the gap between computational complexity and user-friendliness, delivering a versatile tool that supports both research and educational applications in bioinformatics.

Chapter 1

Introduction

Pattern matching is a fundamental problem in computer science that involves finding specific patterns or sequences within a larger body of text. In recent years, the need for efficient pattern-matching algorithms has expanded far beyond traditional text processing — especially in fields like bioinformatics, where vast genomic datasets must be analyzed to identify biologically meaningful sequences known as motifs. Motifs in DNA, RNA, or protein sequences often represent functional or regulatory elements, such as promoter regions, transcription factor binding sites, or conserved domains. Detecting these motifs accurately and efficiently is therefore essential for understanding genetic structure and function.

In this project, we apply concepts from Formal Language and Automata Theory to perform motif detection in biological sequences. Finite automata, particularly Deterministic (DFA) and Nondeterministic Finite Automata (NFA), are mathematical models of computation that can recognize patterns defined over a specific alphabet. Their predictable and well-defined structure makes them ideal for designing algorithms that systematically process large strings of data. By representing motifs as regular

languages, finite automata can efficiently scan long sequences to locate every position where a motif appears.

In this project, motifs expressed using IUPAC nucleotide codes or bracketed character classes are first parsed into symbol sets. A corresponding NFA is constructed to represent all possible transitions through the motif, which is then converted into a DFA using the subset construction algorithm. The resulting DFA serves as a high-speed matcher that can traverse biological sequences symbol by symbol to identify valid motif occurrences. This deterministic approach ensures efficient pattern recognition even across very large input sequences.

To enhance accessibility and understanding, the project includes an interactive web application built using Flask and Graphviz. This interface allows users to input motifs, upload FASTA sequence files, and visualize the automata (NFA and DFA) that represent those motifs. It also provides analytical outputs such as the number of matches, motif positions, and highlighted sequence fragments, combining theoretical computation with practical visualization.

1.1 Problem Statement

Efficient pattern matching is a core problem in computer science, essential for applications such as text processing, information retrieval, and data analysis. Traditional methods like the naive algorithm are computationally expensive for large datasets, while advanced algorithms such as Knuth–Morris–Pratt (KMP) or Boyer–Moore are optimized for specific cases but lack flexibility for handling complex or ambiguous pat-

terns. There is a need for scalable and deterministic approaches that can efficiently identify all pattern occurrences in large or streaming data. This project addresses this challenge by implementing finite automata–based pattern matching, leveraging NFA and DFA for efficient and interpretable pattern recognition. As an application, the technique is demonstrated on biological sequence data for motif detection.

1.2 Objectives

- To implement pattern matching using **Finite Automata (NFA and DFA)** for efficient and deterministic text analysis.
- To convert motifs or patterns into formal automata representations using subset construction.
- To design a system capable of handling large text inputs with improved speed and accuracy.
- To visualize the constructed NFA and DFA for better interpretability.
- To demonstrate the practical application of automata-based pattern matching in **bioinformatics motif searching.**

Chapter 2

Methodology

The project applies Finite Automata Theory to perform motif detection in DNA sequences. Motifs are represented as regular languages, and their occurrences within biological sequences are identified using deterministic computational models. The methodology integrates the principles of automata construction, state transition modeling and deterministic scanning to achieve accurate and efficient pattern recognition.

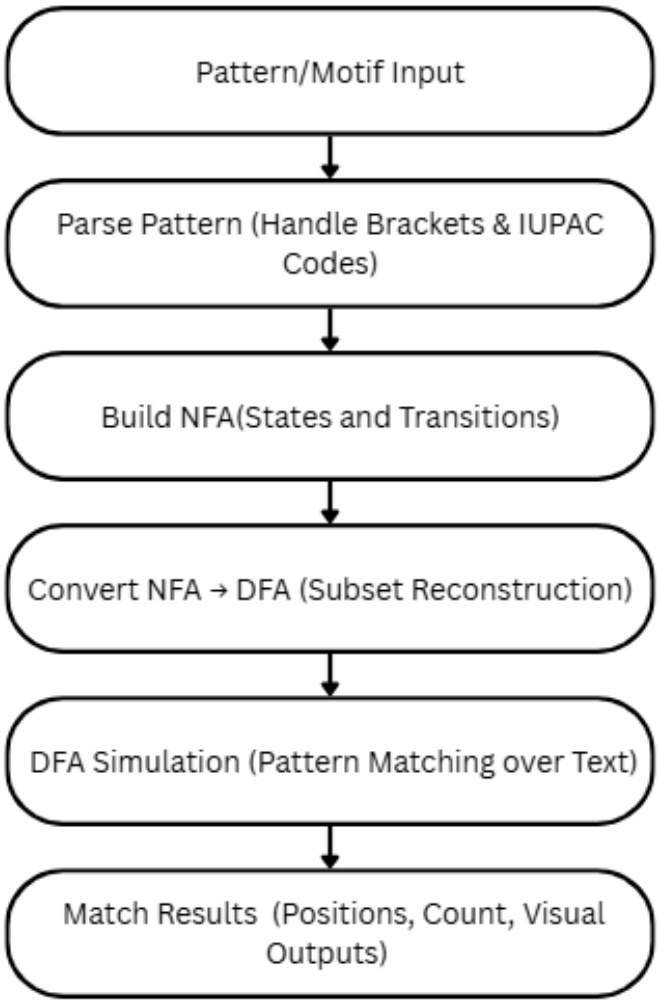


Figure 2.1: Methodology Flowchart for Automata-Based Pattern Matching System

Each step represents a stage in the pattern matching process:

- **Pattern/Motif Input:** The user provides the motif or text pattern to be searched.
- **Parse Pattern:** The system interprets IUPAC codes and bracket classes to identify all valid symbols for each motif position.

- **Build NFA:** A Nondeterministic Finite Automaton is constructed with states and transitions for each valid symbol set.
- **Convert NFA → DFA:** Using subset construction, the NFA is converted into an equivalent Deterministic Finite Automaton for deterministic execution.
- **DFA Simulation:** The DFA scans the input sequence symbol by symbol to detect matches.
- **Match Results:** Outputs positions, counts, and visual representations of matching sequences and automata.

2.1 Background

Motif detection in DNA sequences can be framed as a pattern recognition problem. In this context, each DNA motif represents a set of acceptable symbol combinations (strings) over the alphabet A, C, G, T. To recognize these patterns formally, finite automata are employed:

- A Nondeterministic Finite Automaton (NFA) allows multiple possible transitions for a given input symbol, representing ambiguity and variation within motifs.
- A Deterministic Finite Automaton (DFA), on the other hand, has only one possible transition for each input symbol, making it efficient for real-time sequence scanning.

2.2 Construction of Nondeterministic Finite Automaton (NFA)

The first computational stage involves constructing an NFA to represent all possible paths through the motif. Each state in the NFA corresponds to a position in the motif, and transitions are labeled with sets of allowed nucleotides.

Steps in NFA Construction:

- **State Creation:**

  For a motif of length n, the NFA contains n + 1 states.

  The initial state represents the start of the motif, and the final state represents successful motif recognition.

- **Transition Definition:**

  For each position i in the motif, transitions are created from state i to i + 1 labeled with all nucleotides allowed at that position.

  If the motif contains ambiguous codes (e.g., N = A, C, G, T), the NFA includes parallel transitions for each possible nucleotide.

- **Acceptance Criteria:**

  The final state is designated as the accepting state, indicating a complete match of the motif in the sequence.

The NFA effectively captures all valid variations of the motif, making it suitable for handling complex biological patterns.

2.3 Conversion of NFA to DFA (Subset Construction)

While the NFA can represent multiple possible paths simultaneously, it is not directly efficient for scanning long sequences. Hence, the NFA is transformed into an equivalent Deterministic Finite Automaton (DFA) using the subset construction algorithm.

Key Steps in Subset Construction:

- **State Combination:**

  Each DFA state represents a set of NFA states reachable under specific input conditions.

- **Transition Determinism:**

  For each DFA state and input symbol, only one deterministic transition is defined.

- **Acceptance States:**

  TAny DFA state containing the NFA's accepting state becomes an accepting state in the DFA.

This transformation ensures that the resulting DFA has a unique transition for each input symbol, making it suitable for efficient pattern recognition.

2.4 Dataset

The nucleotide sequence dataset used for motif identification was obtained from the National Center for Biotechnology Information (NCBI). The required DNA sequences

were downloaded in FASTA format from the NCBI Nucleotide (GenBank) database.

- The NCBI portal was accessed at https://www.ncbi.nlm.nih.gov/nuccorehttps://www.ncbi.nlm.nih.gov/nuccore.
- The target organism (e.g., *Enterobacteria phage*) was searched using the database search bar.
- The appropriate sequence record was opened and viewed in FASTA format.
- Using the option "Send to → File → Format: FASTA → Create File", the sequence file was downloaded.
- The downloaded file (e.g., *Enterobacteria.fasta*) was saved in the project's data directory for further motif pattern analysis.

Chapter 3

Implementation

This chapter describes the detailed implementation of the system that performs pattern matching in large texts using finite automata. The following subsections explain each stage of the process, from motif parsing and sequence handling to automata construction, visualization, and results delivery.

3.1 System Workflow Overview

The implementation consists of several integrated stages — motif parsing, sequence handling, automata construction, pattern matching, visualization, and web integration — that together form the end-to-end motif analysis pipeline.

3.1.1 Motif Input and Parsing

- Users input DNA motifs potentially containing ambiguous nucleotide codes based on IUPAC nomenclature.
- The motif parser analyzes the motif string, identifying bracket classes and ambiguous codes.

- Converts the motif into a list of symbol sets, where each set contains all permissible nucleotides for that motif position.

3.1.2 Sequence Data Handling

- Users upload DNA sequences in FASTA format.
- The system reads and cleanses sequences by removing whitespace and standardizing to uppercase for consistency in matching.

3.1.3 Automata Construction

- Constructs a nondeterministic finite automaton (NFA) from symbol sets; each motif position corresponds to states connected by transitions labeled with allowed nucleotides.
- Converts the NFA into an equivalent deterministic finite automaton (DFA) using subset construction algorithms to enable efficient pattern searching.

3.1.4 Pattern Matching

- The DFA is simulated over input sequences character-by-character.
- The search tracks the automaton state traversal, and whenever an accepting state is reached, a motif occurrence is recorded with its start position.

3.1.5 Visualization

- Generates scalable vector graphics (SVG) visualizations of the NFA and DFA to illustrate states and transitions.

- These visualizations help users understand the matching logic and internal automata structures.

3.1.6 Web Application Integration

- Backend implemented in Flask handles file uploads, motif processing, automata creation, and sequence scanning.
- Provides API endpoints for motif matching and automaton graph generation.
- Frontend web interface allows users to upload files, input motifs, initiate searches, and view both textual results and automaton visualizations interactively.

3.1.7 Validation and Results Delivery

- After matching, the system returns detailed results including motif positions in sequences and highlighted matching sequence segments.
- Provides a summary of total sequences analyzed, total matches found, and success percentage.

3.1.8 Extensibility and User Accessibility

- Flexible motif and sequence input formats allow application across diverse biological datasets.
- User-friendly interface encourages non-expert users to leverage complex computational methods.

3.2 Flask 2.3.2

Flask is a lightweight and flexible Python web framework designed for building web applications quickly and with minimal setup. Version 2.3.2 provides easy-to-use routing to map URLs to Python functions, support for handling HTTP requests and responses, built-in development server, and integrated debugging capabilities. Flask follows a modular design, allowing developers to scale up from simple applications to complex ones by adding extensions as needed. It also uses the Jinja2 template engine, enabling dynamic HTML rendering and seamless integration with frontend components. In this project, Flask serves as the backend framework managing HTTP endpoints for file upload, motif processing, sequence analysis, and automata visualization, bridging the frontend user interface with the Python processing logic efficiently.

3.3 Graphviz

Graphviz is an open-source graph visualization software widely used to represent structural information as diagrams of nodes and edges. It provides rendering tools that can create outputs in various formats including SVG, useful in web applications for interactive visualization. In this project, Graphviz is used to generate graphical representations of nondeterministic and deterministic finite automata (NFA and DFA). These visualizations illustrate the states and transitions of the automata, helping users comprehend how motif matching is modeled as state machines. The graphical output from Graphviz is embedded in the web interface to provide clear, informative visuals

that support understanding of the pattern recognition process.

3.4 Frozen Set

A frozen set is an immutable data structure in Python utilized to represent a fixed collection of NFA states. During the conversion of a Nondeterministic Finite Automaton (NFA) into a Deterministic Finite Automaton (DFA), multiple NFA states are combined to form a single DFA state. To ensure that these state combinations remain constant and can be efficiently referenced within transition mappings, they are stored as frozen sets. This approach facilitates reliable state management during the subset construction process.

Chapter 4

Results

This chapter presents the outputs of the developed system for pattern matching using finite automata. The system interface allows users to input motifs, upload FASTA sequences, and visualize both the NFA and DFA generated for the given motif.

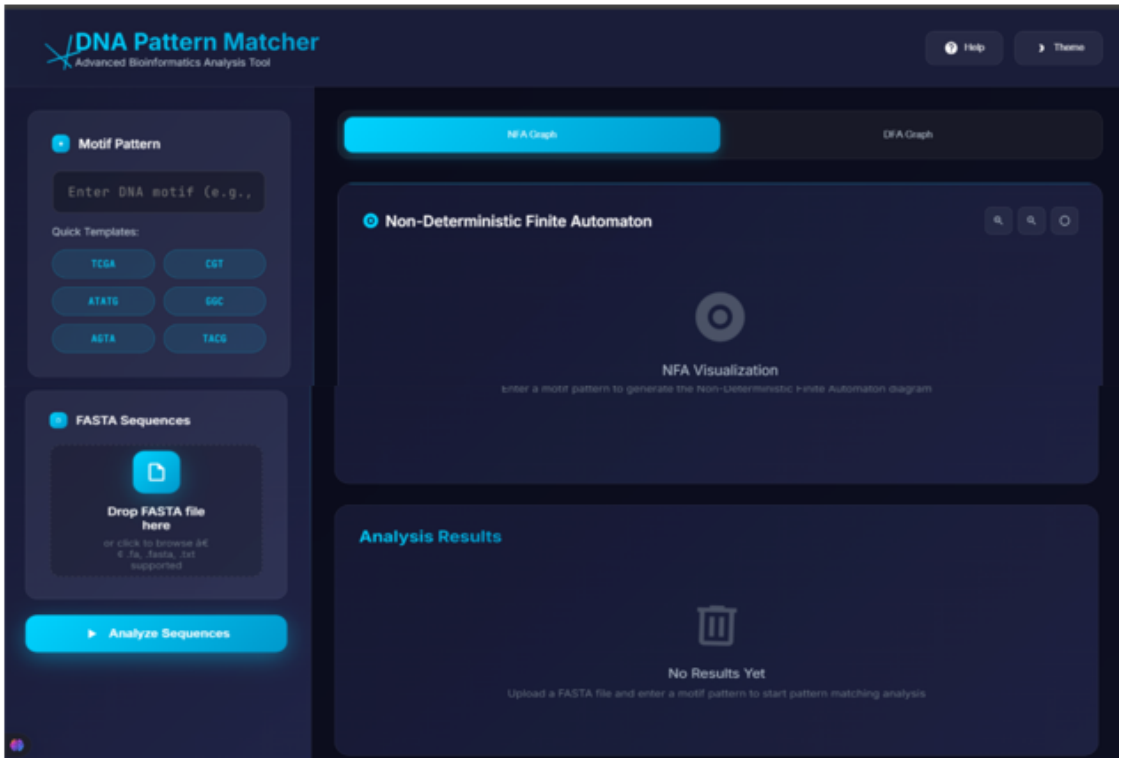


Figure 4.1: Web Page

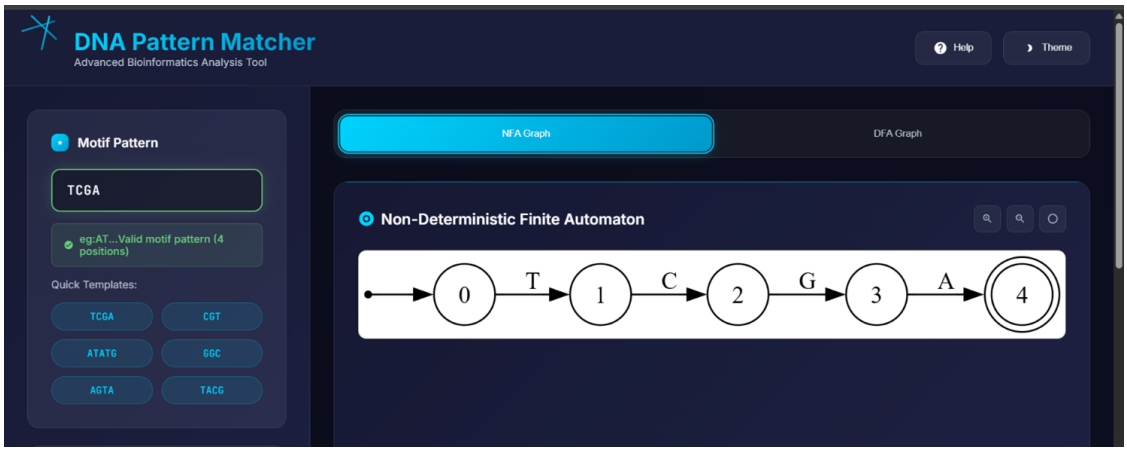


Figure 4.2: NFA Output

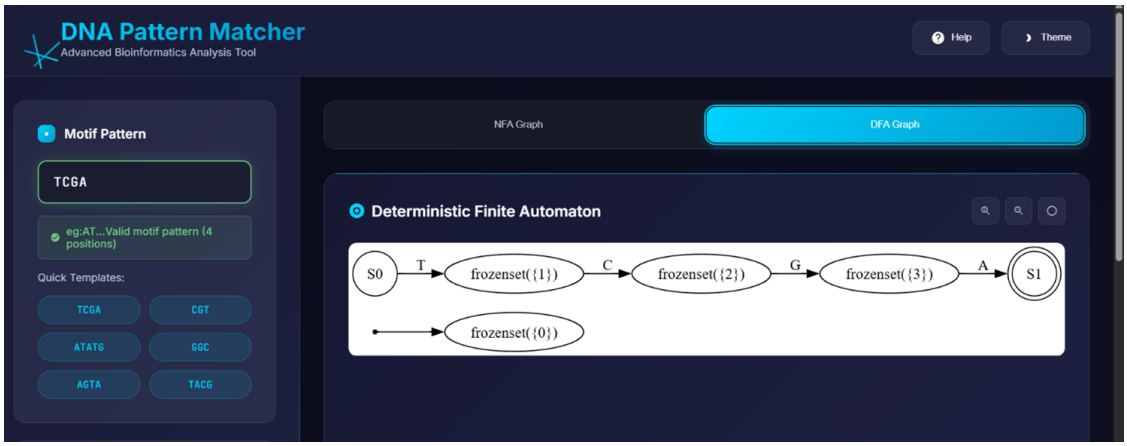


Figure 4.3: DFA Output

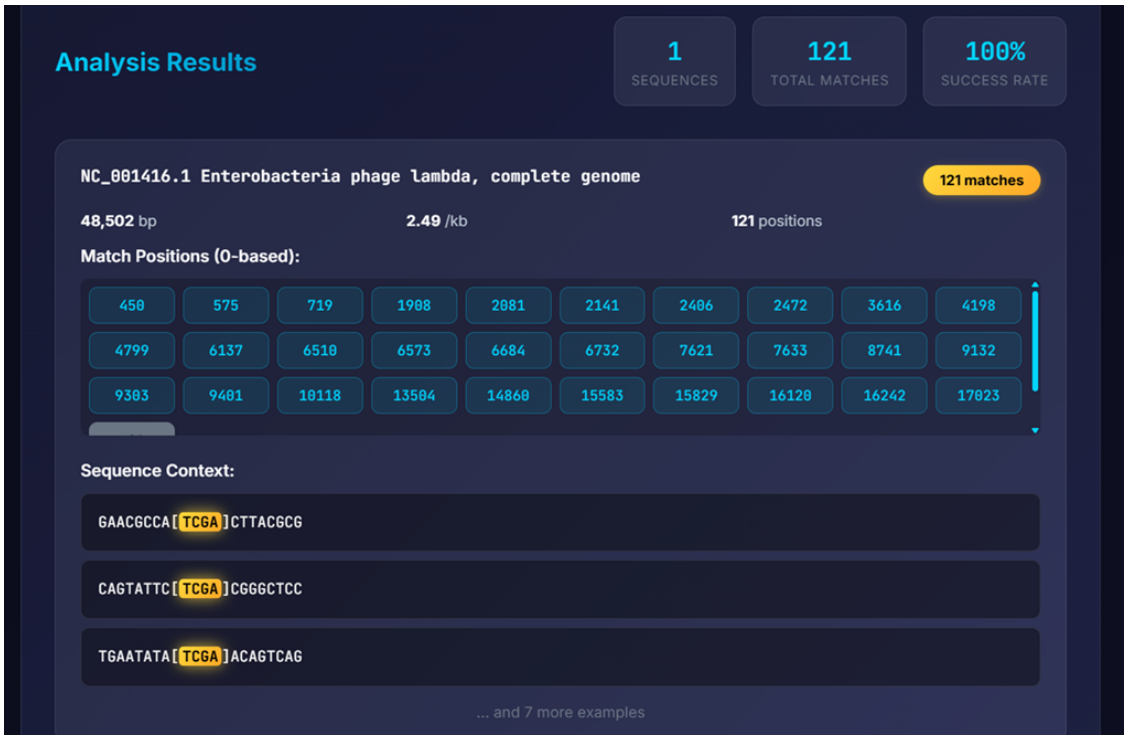


Figure 4.4: Analysis Results

Figure 4.5: Pattern Matching Help Section

Each figure represents a key stage of system functionality:

- **Figure 4.1** shows the web interface used for uploading sequences and inputting motifs.
- **Figure 4.2** displays the generated Nondeterministic Finite Automaton (NFA) for the entered motif.
- **Figure 4.3** shows the corresponding Deterministic Finite Automaton (DFA) after subset construction.
- **Figure 4.4** presents the output statistics, including sequence matches and posi-

tions.

- **Figure 4.5** illustrates the *Pattern Matching Help* section, which guides users on motif syntax, file requirements, and interpretation of results such as positions, density, and automata visualization.

Chapter 5

Discussion

This project successfully demonstrates the integration of automata-based pattern recognition with modern web technologies to perform motif identification in biological sequences. The system utilizes algorithms derived from Finite Automata Theory, where motifs are represented as patterns and processed through Nondeterministic Finite Automata (NFA) and their equivalent Deterministic Finite Automata (DFA) models. This approach enables efficient identification of motif occurrences within large DNA sequences. The backend, developed using Flask, handles motif parsing, NFA/DFA construction, and sequence matching. The logic modules (automata.py, matching.py, and parsing.py) work in unison — parsing motifs into symbol sets, building automata structures, and performing position-wise matching of motifs in uploaded FASTA sequences. The visual representations of NFA and DFA are dynamically generated using Graphviz, allowing users to interpret the motif recognition process visually. The frontend interface (developed using HTML, CSS, and JavaScript) provides an intuitive platform for users to upload FASTA files, enter motif patterns, and visualize the results. Communication between the interface and the backend is achieved via asynchronous HTTP requests,

ensuring real-time interaction and seamless user experience. The generated results include sequence statistics, match positions, and examples that provide biological insight into motif distribution. Overall, the integration of Flask, Graphviz, and interactive frontend visualization led to a cohesive system capable of performing accurate and explainable motif detection. This implementation bridges the gap between theoretical automata concepts and practical bioinformatics applications, making complex motif analysis accessible through an efficient and user-friendly interface.

Chapter 6

Conclusion

This project demonstrates the effective application of Finite Automata Theory to solve the classical problem of pattern matching in large datasets. By representing motifs as formal languages and processing them through Nondeterministic (NFA) and Deterministic Finite Automata (DFA), the system achieves accurate and efficient detection of recurring patterns within biological sequences. The automata-based approach ensures deterministic performance, scalability, and clarity in how matches are recognized, distinguishing it from traditional string-matching algorithms. The successful integration of Flask for backend processing, Graphviz for automata visualization, and a responsive web-based frontend highlights how theoretical computer science concepts can be transformed into practical, interactive tools. Users can not only perform motif searches but also visualize the underlying automata, enhancing both understanding and interpretability of the results. Overall, this project bridges the gap between formal computation theory and real-world data analysis, showing that automata can serve as a powerful foundation for pattern recognition tasks across domains. While the current application focuses on bioinformatics motif detection, the underlying framework can be

extended to other fields such as text mining, network security, and log pattern analysis, making it a versatile and scalable solution for efficient pattern matching.

References

1. Deterministic Finite Automata for pattern matching in FPGA for intrusion detection — *A. BabuKaruppiah, S. Rajaram*, 2011.
   https://ieeexplore.ieee.org/document/5762461/

2. A survey on Finite Automata based pattern matching techniques for network Intrusion Detection System (NIDS) — *P. M. Rathod, N. Marathe, A.V. Vidhate*, 2014.
   https://ieeexplore.ieee.org/abstract/document/7002456

3. O3FA: A Scalable Finite Automata-based Pattern-Matching Engine for Out-of-Order Deep Packet Inspection — *Xiaodong Yu, Wu-chen Feng*, 2016.
   https://dl.acm.org/doi/abs/10.1145/2881025.2881034

4. EfficientPMM: Finite Automata Based Efficient Pattern Matching Machine — *Ramanpreet Singh, Ali A. Ghorbani.*
   https://www.sciencedirect.com/science/article/pii/S1877050917308724

5. Pattern Recognition for Identifying a String Within a Text File Using Finite Automata — *Mamidi Prajana, Harsha Rajkumar, Akepati Sai Sannidhi.*

https://ieeexplore.ieee.org/abstract/document/10723912/