Departamento de CSI
Computación y Estructuras Discretas III
2025-2 Integrative Task 1

Objetivos

Unidad 1 – Lenguajes Regulares y Autómatas

OE1.1 Explicar los conceptos fundamentales de alfabetos, cadenas y lenguajes.

OE1.2 Definir formalmente los distintos tipos de autómatas finitos (deterministas, no deterministas, no deterministas con transiciones lambda, transductores), dar ejemplos, identificar sus elementos y conocer sus aplicaciones.

OE1.3 Reconocer la equivalencia entre expresiones regulares y autómatas finitos.

OE1.4 Entender y aplicar, tanto las nociones de lenguajes y expresiones regulares, como de autómatas de estado finito, para el reconocimiento de patrones, procesamiento, validación y extracción de texto usando un lenguaje de programación.

Unidad 2 – Gramáticas y Lenguajes

OE2.1 Definir una gramática independiente del contexto y construir GICs para lenguajes dados.

OE2.2 Explicar el concepto de ambigüedad lingüística y aplicarlo a GIC ambiguas.

OE2.3 Reconocer la importancia de las formas normales para simplificar GICs.

OE2.4 Aplicar los conceptos de gramáticas generativas para el diseño e implementación de DSLs usando un lenguaje de programación.

OE2.5 Reconocer la importancia de los distintos algoritmos de decisión sobre GIC.

Statement

The objective of this project is to develop a Social Media Content Moderator that processes user-generated posts, detects policy violations (such as hate speech, misinformation, or offensive content), and classifies posts into appropriate categories. The system will leverage formal language theory, including regular expressions, finite automata (FA), finite state transducers (FST), and context-free grammars (CFGs), to parse, analyze, and transform social media text efficiently.

Project goals:

1. Post Preprocessing and Tokenization (Regular Expressions)
    a. **Objective:** Extract and standardize key elements of posts (mentions, hashtags, URLs, emojis).
    b. **Implementation:** Use Python's re module to define regular expressions for identifying user mentions, hashtags, and external links.
    c. **Example:** Detect all hashtags using a regex like r'#\w+' or extract all URLs using r'(https?://\S+)'.
2. Content Moderation (Finite Automata)
    a. **Objective:** Classify posts into categories such as Safe, Needs Review, or Violation based on detected patterns.
    b. **Implementation:** Build a DFA using pyformlang that traverses tokens and recognizes patterns indicating hate speech, offensive language, or spam.
    c. **Example:** DFA moves through states like Start → CheckWords → Violation if it finds forbidden terms.
3. Post Transformation and Action Suggestion (Finite State Transducers)
    a. **Objective:** Suggest actions (e.g., blur image, replace offensive words, send warning) based on classification.
    b. **Implementation:** Use FSTs to transform text, replacing offensive words with *** or generating a moderation message.
    c. **Example:** Post: “You are stupid!” → Transformed: “You are ***!” with a moderation warning.
4. Post Visualizer - Grammar Validation (Context-Free Grammars using textX)

a. **Objective:** Validate the structure of posts for well-formedness and compliance with policy, also produce a well-formed post and display it to the user with at least ten different ways to enhance the post (Formulas in latex is a MUST).
b. **Implementation:** Define a CFG with textX for accepted post formats (mentions, hashtags, text, links), After the validation, produce the code (HTML/Markdown) to show the preview of the post. Possible ehacements, replace some words for emojis, change the look and feel of emojis, present some text in bold, italic, underlined, upsidedown text and math formulas (input in latex)
c. **Example:**
This is a well-formed post :-) with -italic font-, *bold font*, _underlined text_, //text in different font// with the formula $A_1 + B_2 = n^2$

should be displayed:

> This is a well-formed post 😊 with *italic font*, **bold font**, underlined text, //***text in different font***// with the formula $A_1 + B_1 = n^2$

Step-by-Step Program Outline:

- **Text Preprocessing (Regular Expressions):**
  - Detect mentions (@username), hashtags, links, and emojis.
  - Normalize text (lowercasing, removing extra spaces).
- **Content Classification (Finite Automata):**
  - Build DFAs to detect:
    - Hate speech (certain keywords)
    - Offensive language
    - Spam patterns (repeated links or hashtags)
- **Content Transformation (Finite State Transducers):**
  - Replace flagged words with placeholders.
  - Generate moderation suggestions: “Warning: This post violates policy.”
- **Structure Validation (Context-Free Grammars):**
  - Define syntax rules for valid posts:
    Post -> Text [HashtagList] [LinkList]
    HashtagList -> Hashtag | Hashtag HashtagList
    LinkList -> Link | Link LinkList
  - Use textX to parse and validate
  - Include in your grammar the syntax elements to enhance the text, remember that one of those enhancements must be the math formulas originally written in latex.
- **Integration and User Interaction:**
  - Build a web interface or stand-alone tool where users submit posts.
  - Show:
    - Original post
    - Classification result (Safe / Violation)
    - Suggested transformation

    - Validation status
    - Preview of the post visualization

- **Testing and Validation:**
    - Test cases for:
        - Posts with offensive language
        - Posts with multiple hashtags and links
        - Mixed-language posts
    - Unit tests for regex extraction, DFA classification, FST transformations, CFG validation.
- **Documentation and User Guide:**
    - Markdown documentation in docs/ directory.
    - README.md explaining setup, dependencies, and usage.
    - Explanation of grammar and automata design.

Literature Review:
Before implementing your program, you should conduct a review of the literature associated with the problem at hand. This is to guide the project, avoid redundancies, make informed decisions, develop a solid theoretical foundation, and thus obtain results of high impact and relevance.

Deliverables

1. Research poster: [Guide]
2. Design:
    a. Design of modules (functions, inputs-outputs)
    b. Design of test cases, including scenarios.
3. Python implementation:
    a. Complete and correct implementation of the model, UI, and tests

This project can be done in groups of minimum 2 and maximum 3 people.

You should create your repository using the following gitHub classroom invitation [link]. PLease, fill the following form to register your team [link].

Your repository must have at least 10 commits with a difference of at least 2 hours between each of them. These commits must be meaningful in terms of value (e.g., they cannot be simply deleting a variable or changing a method name). In the repository or project, there should be a directory called docs/ where each of the design documents should be placed, documents in your repository should be written using markdown. Include any necessary clarifications for manipulating or understanding your project in the readme.md file as extra documentation (e.g., IDE used, members' names). The contribution of each member will be evaluated based on their input, and the only way to verify this will be through the commits.

Note: The poster as well as the presentation or demonstration showcasing the functionality and features of the system must be in english.

The defense will be done through the presentation of your poster.

Departamento de CSI
Computación y Estructuras Discretas III
2025-2 Integrative Task 1

Deadline for Submission: September 26th