Departamento de CSI
Computación y Estructuras Discretas III
2025-2 Tarea integradora 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 GIC 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 GIC.
OE2.4 Aplicar los conceptos de gramáticas generativas para el diseño e implementación de DSL usando un lenguaje de programación.
OE2.5 Reconocer la importancia de los distintos algoritmos de decisión sobre GIC.

Declaración

El objetivo de este proyecto es desarrollar un Moderador de Contenido para Redes Sociales que procese las publicaciones generadas por los usuarios, detecte infracciones de políticas (como incitación al odio, desinformación o contenido ofensivo) y clasifique las publicaciones en las categorías adecuadas. El sistema aprovechará la teoría del lenguaje formal, incluyendo expresiones regulares, autómatas finitos (AF), transductores de estados finitos (FST) y gramáticas libres de contexto (CFG), para analizar y transformar el texto de las redes sociales de forma eficiente.

Objetivos del proyecto:

1. Pospreprocesamiento y tokenización (expresiones regulares)
    a. **Objetivo:** Extraer y estandarizar elementos clave de las publicaciones (menciones, hashtags, URL, emojis).
    b. **Implementación:** Utilice el módulo re de Python para definir expresiones regulares para identificar menciones de usuarios, hashtags y enlaces externos.
    do. **Ejemplo:** Detecta todos los hashtags usando una expresión regular como r'#\w+' o extrae todas las URL usando r'(https?://\S+)'.
2. Moderación de contenido (autómatas finitos)
    a. **Objetivo:** Clasifique las publicaciones en categorías como Segura, Necesita revisión o Infracción según los patrones detectados.
    b. **Implementación:** Cree un DFA usando pyformlang que recorra tokens y reconozca patrones que indiquen discurso de odio, lenguaje ofensivo o spam.
    do. **Ejemplo:** DFA pasa por estados como Inicio → Verificar palabras → Violación si encuentra términos prohibidos.
3. Transformación posterior y sugerencia de acción (transductores de estados finitos)
    a. **Objetivo:** Sugerir acciones (por ejemplo, desenfocar la imagen, reemplazar palabras ofensivas, enviar advertencia) según la clasificación.
    b. **Implementación:** Utilice FST para transformar texto, reemplazando palabras ofensivas con *** o generando un mensaje de moderación.
    do. **Ejemplo:** Publicación: “¡Eres estúpido!” → Transformado: “¡Eres una mierda!” con una advertencia de moderación.
4. Visualizador de publicaciones: Validación gramatical (Gramáticas libres de contexto con textX)

Departamento de CSI
Computación y Estructuras Discretas III
2025-2 Tarea integradora 1

a. **Objetivo:** Validar la estructura de las publicaciones para verificar que estén bien formadas y que cumplan con las políticas, también producir una publicación bien formada y mostrarla al usuario con al menos diez formas diferentes de mejorar la publicación (las fórmulas en LaTeX son IMPRESCINDIBLES).

b. **Implementación:** Definir un CFG con textX para los formatos de publicación aceptados (menciones, hashtags, texto, enlaces). Tras la validación, generar el código (HTML/Markdown) para mostrar la vista previa de la publicación. Posibles mejoras: reemplazar palabras por emojis, cambiar la apariencia de los emojis, presentar texto en negrita, cursiva, subrayado, texto invertido y fórmulas matemáticas (introducir en LaTeX).

do. **Ejemplo:**

Este es un post bien formado :-) con -fuente cursiva-, *fuente negrita*, _texto subrayado_, //texto en fuente diferente// con la fórmula $A_1 + B_2 = n^2$

Debería mostrarse:

> Esta es una publicación bien formada. con *fuente cursiva*, **fuente en negrita**, texto subrayado yo, //texto en fuente diferente//con la fórmula $A_1 + B_2 = n^2$

Esquema del programa paso a paso:

- **Preprocesamiento de texto (expresiones regulares):**
  - Detecta menciones (@nombreusuario), hashtags, enlaces y emojis.
  - Normalizar el texto (ponerlo en minúsculas, eliminar espacios adicionales).
- **Clasificación de contenido (autómatas finitos):**
  - Crear DFA para detectar:
    - Discurso de odio (determinadas palabras clave)
    - Lenguaje ofensivo
    - Patrones de spam (enlaces o hashtags repetidos)
- **Transformación de contenido (transductores de estados finitos):**
  - Reemplace las palabras marcadas con marcadores de posición.
  - Generar sugerencias de moderación: “Advertencia: esta publicación infringe la política”.
- **Validación de estructura (gramáticas libres de contexto):**
  - Defina reglas de sintaxis para publicaciones válidas:

    Publicación -> Texto [HashtagList] [LinkList] HashtagList -> Hashtag | Hashtag HashtagList LinkList -> Enlace | Enlace LinkList
  - Utilice textX para analizar y validar
  - Incluye en tu gramática los elementos sintácticos para enriquecer el texto, recuerda que una de esas mejoras deben ser las fórmulas matemáticas escritas originalmente en latex.
- **Integración e interacción del usuario:**
  - Construir una interfaz web o una herramienta independiente donde los usuarios envíen publicaciones.
  - Mostrar:
    - Publicación original
    - Resultado de la clasificación (Seguro/Violación)
    - Transformación sugerida

- Estado de validación
- Vista previa de la visualización de la publicación

- **Pruebas y validación:**
  - Casos de prueba para:
    - Publicaciones con lenguaje ofensivo
    - Publicaciones con múltiples hashtags y enlaces
    - Publicaciones en idiomas mixtos
  - Pruebas unitarias para extracción de expresiones regulares, clasificación DFA, transformaciones FST, validación CFG.
- **Documentación y guía del usuario:**
  - Documentación de Markdown en el directorio docs/.
  - README.md explica la configuración, las dependencias y el uso.
  - Explicación de la gramática y diseño de autómatas.

Revisión de literatura:
Antes de implementar su programa, debe realizar una revisión de la literatura relacionada con el problema en cuestión. Esto sirve para guiar el proyecto, evitar redundancias, tomar decisiones informadas, desarrollar una base teórica sólida y, así, obtener resultados de alto impacto y relevancia.

Entregables

1.Póster de investigación: [Guía]

2.Diseño:

a. Diseño de módulos (funciones, entradas y salidas)

b.Diseño de casos de prueba, incluyendo escenarios.

3.Implementación de Python:

a.Implementación completa y correcta del modelo, la interfaz de usuario y las pruebas.

Este proyecto se puede realizar en grupos de mínimø y máximo de pe3onas.

Debes crear tu repositorio utilizando la siguiente invitación de aula de GitHub [enlace]. Por favor, rellene el siguiente formulario para registrar a su equipo [enlace].

Su repositorio debe tener al menos 10 confirmaciones con una diferencia de al menos 2 horas entre cada uno de ellos. Estas confirmaciones deben tener un valor significativo (p. ej., no pueden ser simplemente eliminar una variable o cambiar el nombre de un método). En el repositorio o proyecto, debe haber un directorio llamado docs/ donde se guarden todos los documentos de diseño. Los documentos de su repositorio deben estar escritos en Markdown. Incluya cualquier aclaración necesaria para manipular o comprender su proyecto en el archivo readme.md como documentación adicional (p. ej., IDE utilizado, nombres de los miembros). La contribución de cada miembro se evaluará en función de sus aportaciones, y la única forma de verificarlo será a través de las confirmaciones.

Nota: Tanto el cartel como la presentación o demostración que muestre la funcionalidad y características del sistema deberán estar en inglés.

La defensa se realizará mediante la presentación de su poster.

Departamento de CSI
Computación y Estructuras Discretas III
2025-2 Tarea integradora 1

Fecha límite de presentación:26 de septiembre