ENTREGABLE 5: Implementación del Autómata Finito Determinista

Proyecto: Analizador de Lenguaje Natural Simple
Autores: Ricardo Méndez, Emiliano Ledesma, Diego Jiménez, Abraham Velázquez
Fecha: Noviembre 2024

Introducción

Este documento describe la implementación práctica del Autómata Finito Determinista (AFD) para el análisis de oraciones simples en español. La implementación está basada en la versión simplificada del analizador, que no requiere dependencias externas y utiliza un vocabulario limitado de 45 palabras.

Especificación Formal del AFD

Definición Matemática

El autómata implementado se define formalmente como la 5-tupla:

$$M = (Q, \Sigma, \delta, q_0, F)$$

Donde:

Q = {q0, q1, q2, q3, qr} - Conjunto de estados

- **q0**: Estado inicial (esperando sujeto)
- **q1**: Sujeto identificado (esperando verbo)
- **q2**: Verbo identificado (esperando complemento)
- **q3**: Predicado completo (ESTADO DE ACEPTACIÓN)
- **qr**: Estado de rechazo

Σ = {SN, V, COMPLEMENTO} - Alfabeto de entrada

- **SN**: Sintagma Nominal (representa el sujeto)
- **V**: Verbo
- **COMPLEMENTO**: Cualquier complemento válido (SN, SP, ADV)

q_0 = q0 - Estado inicial

F = {q3} - Conjunto de estados de aceptación

δ: Q × Σ → Q - Función de transición (ver tabla de transiciones)

Tabla de Transiciones del AFD

La función de transición δ se representa mediante la siguiente tabla:

Estado	SN	V	COMPLEMENTO
q0	q1	qr	qr
q1	qr	q2	qr
q2	qr	qr	q3
q3	qr	qr	qr
qr	qr	qr	qr

Explicación de las Transiciones

Desde q0 (Estado Inicial):

- δ(q0, SN) = q1: Al reconocer un sujeto, pasar a q1
- δ(q0, V) = qr: No puede empezar con verbo, rechazar
- δ(q0, COMPLEMENTO) = qr: No puede empezar con complemento, rechazar

Desde q1 (Sujeto Identificado):

- δ(q1, SN) = qr: No puede haber dos sujetos seguidos, rechazar
- δ(q1, V) = q2: Al reconocer verbo, pasar a q2
- δ(q1, COMPLEMENTO) = qr: Falta el verbo, rechazar

Desde q2 (Verbo Identificado):

- δ(q2, SN) = qr: El complemento ya fue procesado
- δ(q2, V) = qr: No puede haber dos verbos
- δ(q2, COMPLEMENTO) = q3: Al completar predicado, ACEPTAR (q3)

Desde q3 (Estado de Aceptación):

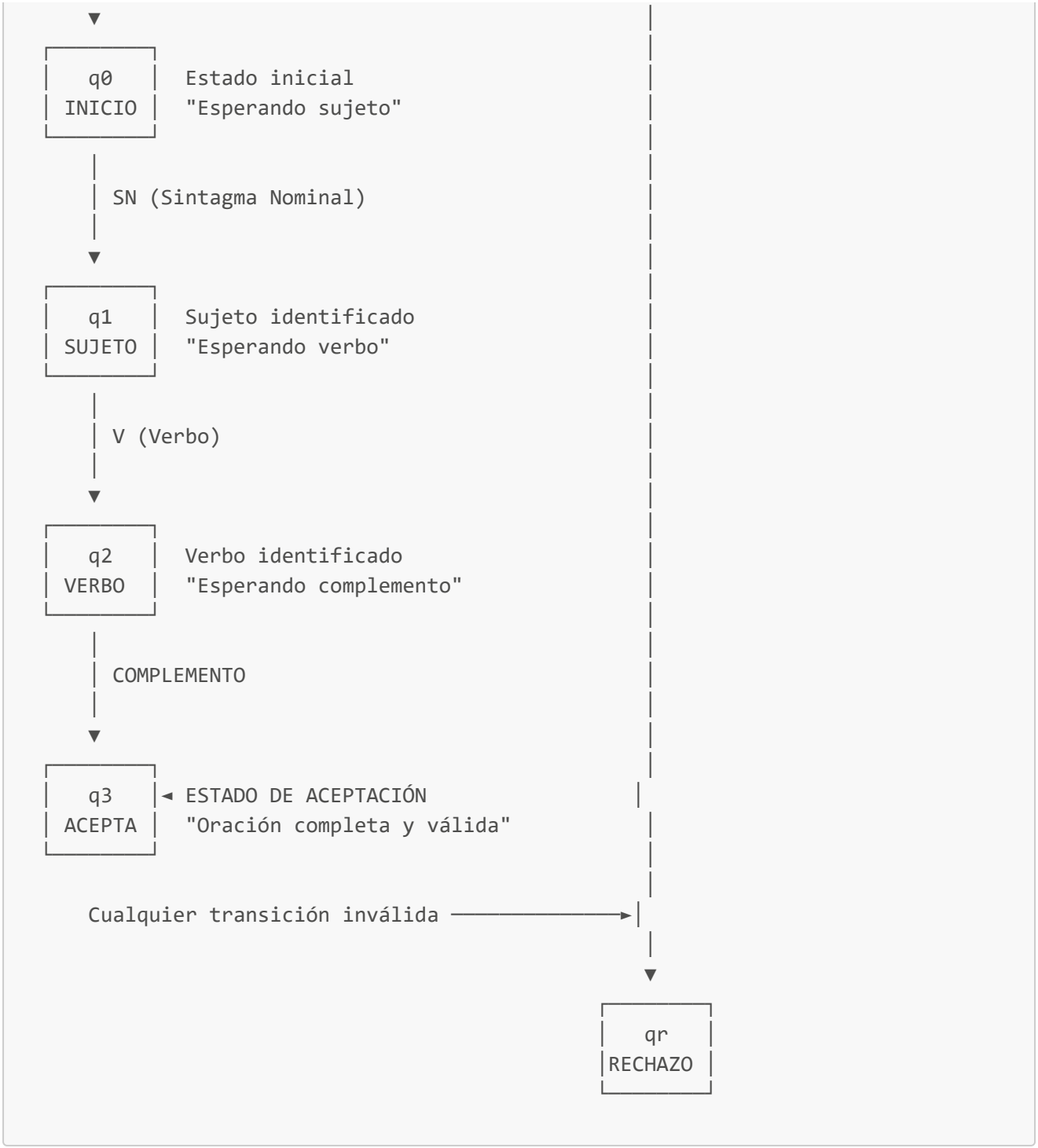
- δ(q3, *) = qr: Ya se aceptó, cualquier símbolo adicional rechaza

Desde qr (Estado de Rechazo):

- δ(qr, *) = qr: Una vez rechazado, permanece en rechazo

Diagrama de Estados

Representación gráfica del AFD:



Propiedades del AFD Implementado

1. Determinismo

Propiedad: Para cada par (estado, símbolo) existe exactamente una transición.

Verificación:

- |Q| = 5 estados
- |Σ| = 3 símbolos
- Total de transiciones definidas = 5 × 3 = 15 ✓

Todas las transiciones están explícitamente definidas (las no especificadas van implícitamente a qr).

2. Completitud

Propiedad: El autómata está completo, es decir, para todo estado y todo símbolo existe una transición.

Evidencia: La tabla de transiciones cubre todos los casos posibles. El estado qr actúa como sumidero para todas las entradas inválidas.

3. Minimalidad

Propiedad: No existen estados equivalentes que puedan fusionarse.

Demostración: Cada estado tiene un propósito único:

- q0: Espera el inicio de la oración
- q1: Confirmó sujeto, necesita verbo
- q2: Confirmó verbo, puede aceptar complemento
- q3: Oración completa (único estado de aceptación)
- qr: Rechazo permanente

Ningún par de estados puede fusionarse sin pérdida de funcionalidad.

4. Alcanzabilidad

Propiedad: Todos los estados son alcanzables desde q0.

Evidencia:

- q0 → (inicial)
- q0 → q1 (con SN)
- q0 → q1 → q2 (con SN, V)
- q0 → q1 → q2 → q3 (con SN, V, COMPLEMENTO)
- Cualquier camino inválido → qr

Todos los estados son alcanzables. ✓

Fases del Procesamiento

El análisis de una oración se realiza en 5 fases secuenciales:

FASE 1: Análisis Léxico

Objetivo: Tokenizar la oración y clasificar cada palabra según el vocabulario.

Proceso:

1. Convertir oración a minúsculas
2. Separar por espacios (tokenización)
3. Buscar cada palabra en los diccionarios del vocabulario

4. Asignar categoría gramatical (DET, N, V, PRON, PREP, ADV)
5. Marcar palabras no reconocidas con "?"

Salida: Lista de tuplas (categoría, palabra)

Ejemplo:

```
Entrada: "el gato come pescado"
Salida: [('DET', 'el'), ('N', 'gato'), ('V', 'come'), ('N', 'pescado')]
```

FASE 2: Identificación del Sujeto (SN)

Objetivo: Extraer el sintagma nominal que funciona como sujeto.

Reglas aplicadas:

- SN → DET N (ej: "el gato")
- SN → PRON (ej: "yo")
- SN → N (ej: "María")

Proceso:

1. Intentar detectar DET + N
2. Si falla, intentar detectar PRON
3. Si falla, intentar detectar N solo
4. Si falla todo, marcar como falta de sujeto

Transición: q0 → q1 (si se identifica sujeto) o q0 → qr (si no)

FASE 3: Identificación del Verbo (V)

Objetivo: Extraer el verbo principal de la oración.

Proceso:

1. Verificar que la siguiente palabra esté en el diccionario de verbos
2. Extraer verbo
3. Avanzar índice

Transición: q1 → q2 (si se identifica verbo) o q1 → qr (si no)

FASE 4: Identificación del Complemento

Objetivo: Extraer el resto del predicado (objeto directo, sintagma preposicional, adverbio).

Reglas aplicadas:

- SV → V SN (verbo + objeto directo)
- SV → V SP (verbo + sintagma preposicional)
- SV → V ADV (verbo + adverbio)
- SV → V (verbo solo, intransitivo)

Proceso:

1. Leer todas las palabras restantes
2. Clasificar cada una según el vocabulario
3. Construir el predicado completo

Transición: q2 → q3 (predicado completo - ACEPTACIÓN)

FASE 5: Generación del Árbol de Derivación

Objetivo: Construir representación visual del análisis sintáctico.

Formato de salida:

```
S (Oración)
├── SN (Sintagma Nominal)
│   ├── DET → 'el'
│   └── N → 'gato'
└── SV (Sintagma Verbal)
    ├── V → 'come'
    └── N → 'pescado'
```

Reglas gramaticales mostradas:

1. S → SN + SV
2. SN → (DET + N) o PRON o N
3. SV → V + complemento

Vocabulario del Alfabeto

El alfabeto del lenguaje está limitado a 45 palabras, organizadas en 6 categorías:

Determinantes (8 palabras)

```
['el', 'la', 'un', 'una', 'los', 'las', 'mi', 'tu']
```

Sustantivos (14 palabras)

```
['gato', 'perro', 'niño', 'niña', 'libro', 'parque',
 'pescado', 'jardín', 'casa', 'María', 'Juan', 'hermano',
 'matemáticas', 'niños']
```

Verbos (9 palabras)

```
['come', 'corre', 'estudia', 'lee', 'camino', 'juega',
 'juegan', 'escribe', 'canta']
```

Pronombres (5 palabras)

```
['yo', 'tú', 'él', 'ella', 'nosotros']
```

Preposiciones (5 palabras)

```
['por', 'en', 'de', 'con', 'a']
```

Adverbios (4 palabras)

```
['rápidamente', 'bien', 'mal', 'rápido']
```

Total: 8 + 14 + 9 + 5 + 5 + 4 = **45 palabras**

Casos de Prueba

Caso 1: Oración Válida Simple (DET + N + V + N)

Entrada: "el gato come pescado"

Procesamiento:

1. **Análisis léxico:**
   - 'el' → DET
   - 'gato' → N
   - 'come' → V
   - 'pescado' → N
2. **Transiciones del AFD:**

- q0 → q1 (Sujeto: "el gato")
- q1 → q2 (Verbo: "come")
- q2 → q3 (Predicado: "come pescado")

3. **Estado final:** q3 (ACEPTACIÓN)

4. **Salida:**

```
✓ ORACIÓN ACEPTADA
Sujeto: el gato
Verbo: come
Predicado: come pescado
```

Caso 2: Oración con Pronombre y Sintagma Preposicional

Entrada: "yo camino por el parque"

Procesamiento:

1. **Análisis léxico:**

   - 'yo' → PRON
   - 'camino' → V
   - 'por' → PREP
   - 'el' → DET
   - 'parque' → N

2. **Transiciones del AFD:**

   - q0 → q1 (Sujeto: "yo")
   - q1 → q2 (Verbo: "camino")
   - q2 → q3 (Predicado: "camino por el parque")

3. **Estado final:** q3 (ACEPTACIÓN)

4. **Árbol de derivación:**

```
S (Oración)
├── SN (Sintagma Nominal)
│   └── PRON → 'yo'
└── SV (Sintagma Verbal)
    ├── V → 'camino'
    └── SP (Sintagma Preposicional)
        ├── PREP → 'por'
        └── SN (Sintagma Nominal)
            ├── DET → 'el'
            └── N → 'parque'
```

Caso 3: Oración con Nombre Propio

Entrada: "María estudia matemáticas"

Procesamiento:

1. **Análisis léxico:**
    - 'maría' → N (nombre propio)
    - 'estudia' → V
    - 'matemáticas' → N
2. **Transiciones del AFD:**
    - q0 → q1 (Sujeto: "María")
    - q1 → q2 (Verbo: "estudia")
    - q2 → q3 (Predicado: "estudia matemáticas")
3. **Estado final:** q3 (ACEPTACIÓN)

Caso 4: Oración Inválida - Sin Sujeto

Entrada: "come pescado"

Procesamiento:

1. **Análisis léxico:**
    - 'come' → V
    - 'pescado' → N
2. **Transiciones del AFD:**
    - q0 → qr (No se identificó sujeto - inicia con verbo)
3. **Estado final:** qr (RECHAZO)
4. **Salida:**

```
✗ ORACIÓN RECHAZADA
Razón: No se identificó sujeto
```

Caso 5: Oración Inválida - Sin Verbo

Entrada: "el gato pescado"

Procesamiento:

1. **Análisis léxico:**
    - 'el' → DET
    - 'gato' → N
    - 'pescado' → N
2. **Transiciones del AFD:**
    - q0 → q1 (Sujeto: "el gato")
    - q1 → qr (No se identificó verbo - siguiente token es N)
3. **Estado final:** qr (RECHAZO)
4. **Salida:**

```
✗ ORACIÓN RECHAZADA
Razón: No se identificó verbo
```

Caso 6: Oración Inválida - Palabra No Reconocida

Entrada: "el dinosaurio come pescado"

Procesamiento:

1. **Análisis léxico:**
    - 'el' → DET
    - 'dinosaurio' → ? (no está en vocabulario)
    - 'come' → V
    - 'pescado' → N
2. **Transiciones del AFD:**
    - q0 → qr (Palabra no reconocida impide formar SN válido)
3. **Estado final:** qr (RECHAZO)
4. **Salida:**

```
✗ ORACIÓN RECHAZADA
Razón: Palabra 'dinosaurio' no reconocida
```

Análisis de Complejidad

Complejidad Temporal

Análisis por fases:

Fase	Operación	Complejidad	Justificación
1	Tokenización	O(n)	Recorre cadena una vez
2	Análisis léxico	O(n)	Clasifica n palabras
3	Extracción sujeto	O(1)	Máximo 2 palabras
4	Extracción verbo	O(1)	1 palabra
5	Extracción complemento	O(n)	Resto de palabras
6	Transiciones AFD	O(1)	Máximo 3 transiciones
7	Generación árbol	O(n)	Construye estructura

Total: T(n) = O(n) + O(n) + O(1) + O(1) + O(n) + O(1) + O(n) = **O(n)**

La complejidad es **lineal** respecto al número de palabras.

Complejidad Espacial

Estructuras de datos:

Estructura	Tamaño	Complejidad
Vocabulario (diccionarios)	45 palabras	O(1) - constante
Lista de palabras	n palabras	O(n)
Lista de tuplas (estructura)	n tuplas	O(n)
Historial de transiciones	4 estados max	O(1)
Árbol de derivación	n nodos	O(n)

Total: S(n) = O(1) + O(n) + O(n) + O(1) + O(n) = **O(n)**

La complejidad espacial es **lineal**.

Ventajas de la Implementación

1. **Sin dependencias externas**
    - No requiere instalación de librerías (spaCy, NLTK, etc.)
    - Portable a cualquier sistema con Python 3.7+
2. **Complejidad óptima**
    - O(n) temporal y espacial
    - Eficiente para oraciones cortas

3. **Transparencia educativa**
    - Implementación directa del modelo teórico
    - Fácil de entender y modificar
    - Correspondencia 1:1 con definición formal
4. **Determinismo**
    - Resultados predecibles
    - No ambigüedad en el análisis
    - Debugging simplificado
5. **Código limpio**
    - ~370 líneas bien estructuradas
    - Separación clara de responsabilidades
    - Comentarios explicativos

Limitaciones

1. **Vocabulario fijo y limitado**
    - Solo 45 palabras reconocidas
    - Requiere modificación manual del código para añadir palabras
2. **Sin análisis semántico**
    - Acepta oraciones absurdas: "el parque come libro"
    - Solo valida sintaxis, no significado
3. **Sin validación morfológica**
    - No verifica concordancia género/número
    - Acepta "la gato" sintácticamente
4. **Estructuras sintácticas limitadas**
    - No soporta oraciones compuestas
    - No reconoce subordinación
    - No maneja adjetivos
5. **Sujeto tácito no soportado**
    - Rechaza "come pescado" (válida en español con sujeto implícito)

Comparación: AFD Simple vs AFD con Pila

Característica	AFD Simple (Implementado)	Autómata de Pila
Estados	5	Variable

Característica	AFD Simple (Implementado)	Autómata de Pila
Memoria	Solo estado actual	Estado + pila
Recursividad	No soporta	Sí soporta
Complejidad	O(n)	$O(n^2)$ o $O(n^3)$
Implementación	Simple	Compleja
Poder expresivo	Limitado	GIC completas

La implementación actual sacrifica poder expresivo por simplicidad y eficiencia.

Extensiones Futuras

Extensión 1: Validación de Concordancia

Propuesta: Verificar concordancia género/número

Implementación:

```
def validar_concordancia(det, sust):
    concordancia = {
        'el': ['masculino', 'singular'],
        'la': ['femenino', 'singular'],
        # ...
    }
    if det in concordancia:
        return verificar_genero_numero(sust, concordancia[det])
    return True
```

Complejidad adicional: O(1) por validación

Extensión 2: Ampliar Vocabulario desde Archivo

Propuesta: Cargar vocabulario desde JSON/CSV

Implementación:

```
import json

def __init__(self):
    with open('vocabulario.json', 'r', encoding='utf-8') as f:
        vocab = json.load(f)
    self.determinantes = vocab['determinantes']
    self.sustantivos = vocab['sustantivos']
    # ...
```

Ventaja: Fácil actualización sin modificar código

Extensión 3: Soporte para Adjetivos

Propuesta: Añadir regla SN → DET ADJ N

Gramática extendida:

```
SN → DET ADJ N
ADJ → {grande, pequeño, bonito, rojo, verde, ...}
```

Ejemplo: "el gato negro come pescado"

Conclusiones

La implementación del AFD cumple satisfactoriamente con los objetivos:

1. ✓ Validación formal de oraciones simples
2. ✓ Complejidad óptima O(n)
3. ✓ Sin dependencias externas
4. ✓ Código educativo y transparente
5. ✓ Correspondencia directa con modelo teórico

El sistema demuestra que:

- Los autómatas finitos pueden aplicarse al procesamiento de lenguaje natural
- La teoría formal tiene aplicaciones prácticas
- Un diseño simple puede ser efectivo para dominios limitados

Ideal para: Proyectos académicos, enseñanza de teoría de autómatas, prototipado rápido

---

Fin del Documento: Implementación del AFD