Referencia de API - Versión Simplificada

Documentación de Clases y Métodos

Archivo: `version_simplificada.py`
Versión: 1.0
Fecha: Noviembre 2024

Índice

Clase AnalizadorSimple

Descripción General

Analizador léxico simple basado en diccionarios de palabras predefinidas. Identifica categorías gramaticales mediante búsqueda en listas.

Constructor

```
def __init__(self)
```

Descripción: Inicializa el analizador con los diccionarios de vocabulario.

Parámetros: Ninguno

Atributos inicializados:

Atributo	Tipo	Descripción	Tamaño
`determinantes`	`List[str]`	Lista de determinantes	8 palabras
`sustantivos`	`List[str]`	Lista de sustantivos	14 palabras
`verbos`	`List[str]`	Lista de verbos	9 palabras
`pronombres`	`List[str]`	Lista de pronombres	5 palabras
`preposiciones`	`List[str]`	Lista de preposiciones	5 palabras
`adverbios`	`List[str]`	Lista de adverbios	4 palabras

Ejemplo de uso:

```
analizador = AnalizadorSimple()
print(len(analizador.verbos))  # Output: 9
```

Método analizar_oracion()

```
def analizar_oracion(self, oracion: str) -> dict
```

Descripción: Analiza una oración y extrae sus componentes gramaticales.

Parámetros:

Nombre	Tipo	Requerido	Descripción
`oracion`	`str`	Sí	Oración a analizar

Retorna: `dict` con las siguientes claves:

Clave	Tipo	Descripción
`valida`	`bool`	True si tiene sujeto y verbo
`sujeto`	`str` o `None`	Sintagma nominal identificado
`verbo`	`str` o `None`	Verbo principal
`predicado`	`str` o `None`	Verbo + complementos
`estructura`	`List[Tuple[str, str]]`	Lista de (categoría, palabra)

Complejidad: O(n) donde n = número de palabras

Ejemplo de uso:

```
analizador = AnalizadorSimple()
resultado = analizador.analizar_oracion("el gato come pescado")

print(resultado['valida'])      # True
print(resultado['sujeto'])      # "el gato"
print(resultado['verbo'])       # "come"
print(resultado['predicado'])   # "come pescado"
print(resultado['estructura'])  # [('DET', 'el'), ('N', 'gato'), ...]
```

Casos especiales:

1. **Sujeto con determinante:**

```
analizar_oracion("el gato come")
# sujeto = "el gato" (DET + N)
```

2. **Sujeto pronominal:**

```
analizar_oracion("yo camino")
# sujeto = "yo" (PRON)
```

3. **Nombre propio:**

```
analizar_oracion("María estudia")
# sujeto = "María" (N sin DET)
```

4. **Sin sujeto:**

```
analizar_oracion("por el parque")
# sujeto = None, valida = False
```

Clase AFDSimple

Descripción General

Implementa un Autómata Finito Determinista (AFD) que valida oraciones mediante transiciones de estado.

Constructor

```
def __init__(self)
```

Descripción: Inicializa el AFD con su definición formal y el analizador léxico.

Parámetros: Ninguno

Atributos inicializados:

Atributo	Tipo	Descripción
`analizador`	`AnalizadorSimple`	Instancia del analizador léxico
`Q`	`List[str]`	Conjunto de estados ['q0', 'q1', 'q2', 'q3', 'qr']

Atributo	Tipo	Descripción
`q0`	`str`	Estado inicial 'q0'
`F`	`List[str]`	Estados de aceptación ['q3']
`delta`	`Dict[Tuple, str]`	Función de transición
`estado_actual`	`str`	Estado actual del autómata
`historial`	`List[Tuple]`	Historial de transiciones

Definición formal del AFD:

```
M = (Q, Σ, δ, q₀, F)

Q = {q0, q1, q2, q3, qr}
Σ = {SN, V, COMPLEMENTO}
q₀ = q0
F = {q3}

δ = {
    (q0, SN) → q1
    (q1, V)  → q2
    (q2, COMPLEMENTO) → q3
}
```

Ejemplo de uso:

```
afd = AFDSimple()
print(afd.Q)   # ['q0', 'q1', 'q2', 'q3', 'qr']
print(afd.F)   # ['q3']
```

Método procesar_oracion()

```
def procesar_oracion(self, oracion: str) -> dict
```

Descripción: Procesa una oración a través del autómata y genera el resultado completo.

Parámetros:

Nombre	Tipo	Requerido	Descripción
`oracion`	`str`	Sí	Oración a procesar

Retorna: `dict` con las siguientes claves:

Clave	Tipo	Descripción
`aceptada`	`bool`	True si el estado final está en F
`estado_final`	`str`	Estado final alcanzado ('q3' o 'qr')
`historial`	`List[Tuple[str, str]]`	Lista de (estado, razón)
`analisis`	`dict`	Resultado del análisis léxico

Complejidad: O(n) dominado por el análisis léxico

Flujo de ejecución:

```
1. Resetear autómata (estado = q0)
2. Analizar oración léxicamente
3. Transición q0 → q1 (si hay sujeto)
4. Transición q1 → q2 (si hay verbo)
5. Transición q2 → q3 (si hay complemento)
6. Verificar aceptación (estado_final ∈ F?)
7. Si aceptada: generar árbol de derivación
8. Retornar resultado
```

Ejemplo de uso:

```
afd = AFDSimple()
resultado = afd.procesar_oracion("el gato come pescado")

print(resultado['aceptada'])       # True
print(resultado['estado_final'])   # 'q3'
print(resultado['historial'])      # [('q0', 'Inicio'), ('q1', '...'), ...]

for estado, razon in resultado['historial']:
    print(f"{estado}: {razon}")
```

Salida de ejemplo:

```
======================================================================
VERSIÓN SIMPLIFICADA - ANALIZADOR DE ORACIONES CON AFD
======================================================================

Oración: 'el gato come pescado'

----------------------------------------------------------------------
FASE 1: Análisis léxico
----------------------------------------------------------------------

Palabras reconocidas:
```

```
  'el' → DET
  'gato' → N
  'come' → V
  'pescado' → N

----------------------------------------------------------------------
FASE 2: Transiciones del AFD
----------------------------------------------------------------------

  q0 → q1
  Razón: Sujeto identificado: 'el gato'

  q1 → q2
  Razón: Verbo identificado: 'come'

  q2 → q3
  Razón: Predicado completo: 'come pescado'

======================================================================
RESULTADO FINAL
======================================================================

Estado final: q3
¿Es estado de aceptación?: SÍ

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

----------------------------------------------------------------------
ÁRBOL DE DERIVACIÓN:
----------------------------------------------------------------------

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

----------------------------------------------------------------------
REGLAS GRAMATICALES APLICADAS:
----------------------------------------------------------------------
1. S → SN + SV
2. SN → DET + N
3. SV → V + complemento
```

Método _transicion() (Privado)

```
def _transicion(self, nuevo_estado: str, razon: str) -> None
```

Descripción: Realiza una transición de estado y registra el historial.

Parámetros:

Nombre	Tipo	Descripción
`nuevo_estado`	`str`	Estado destino ('q1', 'q2', 'q3', 'qr')
`razon`	`str`	Descripción de por qué se hace la transición

Retorna: `None` (modifica estado interno)

Efecto secundario:

- Actualiza `self.estado_actual`
- Añade entrada a `self.historial`
- Imprime la transición en consola

Ejemplo interno:

```
# Llamado dentro de procesar_oracion()
self._transicion('q1', "Sujeto identificado: 'el gato'")

# Imprime:
#   q0 → q1
#   Razón: Sujeto identificado: 'el gato'
```

Método _generar_resultado() (Privado)

```
def _generar_resultado(self, analisis: dict, aceptada: bool) -> dict
```

Descripción: Genera el diccionario de resultado final.

Parámetros:

Nombre	Tipo	Descripción
`analisis`	`dict`	Resultado del análisis léxico
`aceptada`	`bool`	Si la oración fue aceptada

Retorna: `dict` con el resultado completo (ver `procesar_oracion()`)

Método _imprimir_arbol_derivacion() (Privado)

```
def _imprimir_arbol_derivacion(self, analisis: dict) -> None
```

Descripción: Imprime el árbol de derivación gramatical en formato ASCII.

Parámetros:

Nombre	Tipo	Descripción
analisis	dict	Resultado del análisis con estructura

Retorna: None (imprime en consola)

Formato de salida:

```
S (Oración)
├── SN (Sintagma Nominal)
│   ├── DET → 'palabra'
│   └── N → 'palabra'
└── SV (Sintagma Verbal)
    ├── V → 'palabra'
    └── COMPLEMENTO → 'palabras...'
```

Algoritmo:

1. Imprimir raíz S
2. Imprimir rama SN con sus componentes
3. Imprimir rama SV con sus componentes
4. Determinar y mostrar reglas aplicadas

Función main()

```
def main() -> None
```

Descripción: Función principal que ejecuta la interfaz interactiva del programa.

Parámetros: Ninguno

Retorna: None

Funcionalidad:

1. Muestra banner de bienvenida
2. Muestra vocabulario disponible

3. Muestra ejemplos de oraciones válidas
4. Inicia loop interactivo:
    - Lee oración del usuario
    - Procesa con el AFD
    - Muestra resultado
    - Repite hasta que el usuario escriba 'salir'

Ejemplo de ejecución:

```
╔═══════════════════════════════════════════════════════════════════╗
║               VERSIÓN SIMPLIFICADA - AFD                          ║
║                    Analizador de Oraciones                        ║
╚═══════════════════════════════════════════════════════════════════╝

Características:
  - Sin dependencias externas (no requiere spaCy)
  - Vocabulario limitado (alfabeto finito)
  - AFD explícito con tabla de transiciones
  - Árbol de derivación ASCII

----------------------------------------------------------------------
VOCABULARIO DISPONIBLE:
----------------------------------------------------------------------
Determinantes: el, la, un, una, los, las, mi, tu
Sustantivos: gato, perro, niño, niña, libro, parque, pescado, ...
Verbos: come, corre, estudia, lee, camino, juega, juegan, escribe, canta
Pronombres: yo, tú, él, ella, nosotros
Preposiciones: por, en, de, con, a

----------------------------------------------------------------------
EJEMPLOS DE ORACIONES VÁLIDAS:
----------------------------------------------------------------------
  - el gato come pescado
  - yo camino por el parque
  - María estudia matemáticas
  - los niños juegan en el jardín

----------------------------------------------------------------------
Ingresa una oración para analizar
(o escribe 'salir' para terminar)
----------------------------------------------------------------------

Oración: _
```

Comandos especiales:

Comando	Descripción
`salir`	Termina el programa

Comando	Descripción
`exit`	Termina el programa
`quit`	Termina el programa

Tipos de Datos

Resultado de Análisis Léxico

```
{
    'valida': bool,                      # Si tiene estructura válida
    'sujeto': str | None,                 # Sintagma nominal
    'verbo': str | None,                  # Verbo principal
    'predicado': str | None,              # Verbo + complementos
    'estructura': List[Tuple[str, str]]  # [(categoría, palabra), ...]
}
```

Ejemplo:

```
{
    'valida': True,
    'sujeto': 'el gato',
    'verbo': 'come',
    'predicado': 'come pescado',
    'estructura': [
        ('DET', 'el'),
        ('N', 'gato'),
        ('V', 'come'),
        ('N', 'pescado')
    ]
}
```

Resultado de Procesamiento AFD

```
{
    'aceptada': bool,                     # Si fue aceptada
    'estado_final': str,                  # 'q3' o 'qr'
    'historial': List[Tuple[str, str]],   # [(estado, razón), ...]
    'analisis': dict                      # Resultado del análisis léxico
}
```

Ejemplo:

```
{
    'aceptada': True,
    'estado_final': 'q3',
    'historial': [
        ('q0', 'Estado inicial'),
        ('q1', "Sujeto identificado: 'el gato'"),
        ('q2', "Verbo identificado: 'come'"),
        ('q3', "Predicado completo: 'come pescado'")
    ],
    'analisis': {
        'valida': True,
        'sujeto': 'el gato',
        'verbo': 'come',
        'predicado': 'come pescado',
        'estructura': [...]
    }
}
```

Categorías Gramaticales

Código	Nombre	Ejemplos
DET	Determinante	el, la, un, una, los, las, mi, tu
N	Sustantivo	gato, perro, niño, María, Juan
V	Verbo	come, corre, estudia, lee, camino
PRON	Pronombre	yo, tú, él, ella, nosotros
PREP	Preposición	por, en, de, con, a
ADV	Adverbio	rápidamente, bien, mal, rápido
?	Desconocido	Palabra no en vocabulario

Constantes

Estados del AFD

```
Q = ['q0', 'q1', 'q2', 'q3', 'qr']

# Descripción de estados:
# q0: Estado inicial (esperando entrada)
# q1: Sujeto identificado
# q2: Verbo identificado
# q3: Predicado completo (ACEPTACIÓN)
# qr: Estado de rechazo
```

Alfabeto de Entrada

```
Σ = {
    'SN': 'Sintagma Nominal (sujeto)',
    'V': 'Verbo',
    'COMPLEMENTO': 'Complemento del verbo'
}
```

Ejemplos de Uso Completos

Ejemplo 1: Uso Básico

```
from version_simplificada import AFDSimple

# Crear instancia
afd = AFDSimple()

# Procesar oración
resultado = afd.procesar_oracion("el gato come pescado")

# Verificar si fue aceptada
if resultado['aceptada']:
    print("✓ Oración válida")
    print(f"Sujeto: {resultado['analisis']['sujeto']}")
    print(f"Verbo: {resultado['analisis']['verbo']}")
else:
    print("✗ Oración inválida")
    print(f"Estado final: {resultado['estado_final']}")
```

Ejemplo 2: Análisis de Múltiples Oraciones

```
from version_simplificada import AFDSimple

afd = AFDSimple()

oraciones = [
    "el gato come pescado",
    "yo camino por el parque",
    "María estudia matemáticas",
    "por el parque",  # Inválida
]

resultados = []
for oracion in oraciones:
    resultado = afd.procesar_oracion(oracion)
```

```
    resultados.append({
        'oracion': oracion,
        'aceptada': resultado['aceptada']
    })

# Mostrar resumen
aceptadas = sum(1 for r in resultados if r['aceptada'])
print(f"\nTotal: {len(oraciones)}")
print(f"Aceptadas: {aceptadas}")
print(f"Rechazadas: {len(oraciones) - aceptadas}")
```

Ejemplo 3: Solo Análisis Léxico

```
from version_simplificada import AnalizadorSimple

analizador = AnalizadorSimple()

# Analizar sin validar con AFD
resultado = analizador.analizar_oracion("el perro corre rápido")

print("Componentes:")
print(f"  Sujeto: {resultado['sujeto']}")
print(f"  Verbo: {resultado['verbo']}")
print(f"  Predicado: {resultado['predicado']}")

print("\nEstructura detallada:")
for categoria, palabra in resultado['estructura']:
    print(f"  {palabra} ({categoria})")
```

Salida:

```
Componentes:
  Sujeto: el perro
  Verbo: corre
  Predicado: corre rápido

Estructura detallada:
  el (DET)
  perro (N)
  corre (V)
  rápido (ADV)
```

Ejemplo 4: Verificar Vocabulario

```
from version_simplificada import AnalizadorSimple

analizador = AnalizadorSimple()

# Ver todas las palabras disponibles
print("Determinantes:", analizador.determinantes)
print("Sustantivos:", analizador.sustantivos)
print("Verbos:", analizador.verbos)

# Verificar si una palabra está en el vocabulario
palabra = "dinosaurio"
if palabra in analizador.sustantivos:
    print(f"'{palabra}' está en el vocabulario")
else:
    print(f"'{palabra}' NO está en el vocabulario")
```

Manejo de Errores

Oraciones Vacías

```
resultado = afd.procesar_oracion("")
# Output: Oración rechazada (sin componentes)
```

Palabras No Reconocidas

```
resultado = afd.procesar_oracion("el dinosaurio come plátano")
# 'dinosaurio' y 'plátano' marcados como '?' en estructura
# Oración probablemente rechazada
```

Estructura Incompleta

```
resultado = afd.procesar_oracion("por el parque")
# Sin sujeto ni verbo
# Estado final: qr (rechazo)
```

Mejores Prácticas

1. Reutilizar Instancias

```

```

```
# ✓ BIEN: Crear una instancia y reutilizarla
afd = AFDSimple()
for oracion in lista_oraciones:
    resultado = afd.procesar_oracion(oracion)

# ✗ MAL: Crear nueva instancia cada vez (ineficiente)
for oracion in lista_oraciones:
    afd = AFDSimple()  # Innecesario
    resultado = afd.procesar_oracion(oracion)
```

2. Verificar Resultado Antes de Acceder

```
resultado = afd.procesar_oracion(oracion)

# ✓ BIEN: Verificar antes de usar
if resultado['aceptada']:
    sujeto = resultado['analisis']['sujeto']
    print(f"Sujeto: {sujeto}")

# ✗ MAL: Asumir que siempre es aceptada
sujeto = resultado['analisis']['sujeto']  # Puede ser None
```

3. Extender Vocabulario Apropiadamente

```
# ✓ BIEN: Extender listas existentes
analizador.sustantivos.extend(['árbol', 'flor', 'sol'])

# ✗ MAL: Reemplazar listas completas
analizador.sustantivos = ['árbol']  # Pierde palabras anteriores
```

Limitaciones Conocidas

1. **Vocabulario finito**: Solo 45 palabras predefinidas
2. **Sin lematización**: No reconoce variaciones (comió ≠ come)
3. **Sin análisis semántico**: Acepta "el parque come libro"
4. **Estructura simple**: No maneja oraciones compuestas
5. **Sensible a mayúsculas**: "El" ≠ "el" (todo se convierte a minúsculas)

Información de Versión

Versión: 1.0
Fecha: Noviembre 2024

Autores: Ricardo Méndez, Emiliano Ledesma, Diego Jiménez y Abraham Velázquez
Licencia: Uso académico

Fin de la Referencia de API