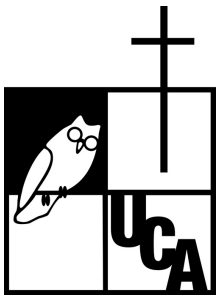

Universidad Centroamericana José Simeón Cañas

Teoría de lenguajes de programación

Catedrático: Jaime Clímaco

Mini-parser para Lenguaje Natural Limitado

Integrante	Carné
Andres Felipe Cardona Duarte	00037820
Axel Jared Hernández Servellón	00145319
Moises Ezequiel Juárez Mejía	00038221
Josue Alfredo Mejia Urias	00000921

Informe Fase 2 - Mini-parser para Lenguaje Natural Limitado

1. Diseño de la Gramática

1.1. Vocabulario Limitado

Se definió un vocabulario de **30 palabras** organizadas en:

• **Determinantes (6)**: el, la, los, las, un, una

• **Sustantivos (10)**: perro, gato, casa, carne, agua, niño, niña, libro, mesa, coche

• **Verbos (8)**: come, bebe, lee, corre, juega, duerme, camina, escribe

• **Adjetivos (9)**: grande, pequeño, rojo, azul, bonito, rápido, lento, nuevo, viejo

• **Puntuación (3)**: ., ?, !

1.2. Gramática Libre de Contexto (CFG)

```
ORACION -> SUJETO VERBO OBJETO [PUNTO] SUJETO -> DETERMINANTE SUSTANTIVO
[ADJETIVO] | SUSTANTIVO [ADJETIVO] | ADJETIVO SUSTANTIVO OBJETO ->
DETERMINANTE SUSTANTIVO [ADJETIVO] | SUSTANTIVO [ADJETIVO] | ADJETIVO
SUSTANTIVO | vacío
```

Características:

• Estructura **SVO (Sujeto-Verbo-Objeto)**

• Adjetivos pueden ir **antes o después** del sustantivo

• Objeto es **opcional** (verbos intransitivos)

• Puntuación **opcional**

2. Implementación

2.1. Parser Descendente Recursivo

Se implementó un parser descendente recursivo en Python (`parser_natural.py`) con las siguientes características:

• **Tokenización**: Convierte texto en tokens según el vocabulario

• **Parsing recursivo**: Cada regla gramatical es una función recursiva

• **Manejo de errores**: Excepciones claras con posición del error

• **Estructura de salida**: Árbol sintáctico en formato diccionario

2.2. Estructura del Código

```
class ParserNatural: - token_actual(): Obtiene token sin consumirlo - consumir(): Consume token si coincide con tipo esperado - parse(): Método principal - parse_oracion(): ORACION -> SUJETO VERBO OBJETO - parse_sujeto(): Implementa reglas de SUJETO - parse_verbo(): Implementa reglas de VERBO - parse_objeto(): Implementa reglas de OBJETO
```

3. Pruebas y Resultados

3.1. Ejemplos Válidos

Oración	Resultado	Estructura
"El perro come carne."	**Aceptada**	**SVO con determinantes**
"Perro come carne."	Aceptada	SVO sin determinante
"El perro grande come carne."	Aceptada	Adjetivo después del sustantivo
"El niño lee libro nuevo."	Aceptada	Adjetivo después del objeto
"El niño corre."	Aceptada	Sin objeto (intransitivo)

3.2. Ejemplos Inválidos

Oración	Resultado	Razón del Error
"Come el perro carne."	**Rechazada**	**Orden incorrecto (VSO)**
"El perro carne."	Rechazada	Falta verbo
"El elefante come hierba."	Rechazada	Palabra fuera del vocabulario
"El perro."	Rechazada	Estructura incompleta

4. Comparación con NLP Moderno (spaCy)

4.1. Robustez

Aspecto	Parser Formal	spaCy
Variaciones de orden	**Rechaza VSO**	**Acepta múltiples órdenes**
Palabras desconocidas	Rechaza	Infiere del contexto
Estructuras flexibles	Solo SVO	Múltiples estructuras
Errores ortográficos	Rechaza	Tolerante

Ejemplo:

• Parser: "Come el perro carne." → Error

• spaCy: "Come el perro carne." → Acepta (interpreta como válida)

4.2. Ambigüedad

Tipo	Parser Formal	spaCy
Ambigüedad léxica	**No resuelve**	**Usa contexto**
Ambigüedad sintáctica	Requiere gramática no ambigua	Resuelve estadísticamente
Ambigüedad semántica	No maneja	Usa embeddings

Ejemplo:

• "El banco está cerrado" (banco = institución o mueble)

• Parser: No diferencia

• spaCy: Usa contexto para desambiguar

4.3. Escalabilidad

Aspecto	Parser Formal	spaCy
Vocabulario	**30 palabras**	**Miles de palabras**
Nuevas palabras	Requiere modificar código	Aprende automáticamente
Diferentes idiomas	Requiere nueva gramática	Modelos pre-entrenados
Dominios específicos	Fácil adaptar	Requiere fine-tuning

Limitación del Parser:

• Solo reconoce palabras del vocabulario predefinido

• Agregar nuevas palabras requiere modificar `VOCABULARIO`

• No puede generalizar a palabras similares

4.4. Aplicabilidad Práctica

Parser Formal - Mejor para:

• **Dominios específicos** con reglas claras

• **Validación estricta** de formato

• **Sistemas embebidos** con recursos limitados

• **Gramáticas de programación** (como en Fase 1)

• **Procesamiento rápido** de estructuras conocidas

spaCy/NLP Moderno - Mejor para:

• **Procesamiento general** de texto

• **Análisis de sentimientos**

• **Extracción de información**

• **Traducción automática**

• **Chatbots y asistentes virtuales**

4.5. Velocidad

Métrica	Parser Formal	spaCy
Tiempo de parsing	**~0.001s**	**~0.01-0.1s**
Memoria	Mínima	~500MB (modelo)
Inicialización	Instantánea	~1-2s (cargar modelo)

Ventaja del Parser: Muy rápido para oraciones válidas, ideal para procesamiento en tiempo real.

5. Análisis de Diferencias

5.1. Ventajas del Parser Formal

1. **Precisión**: Acepta solo estructuras exactas según la gramática
2. **Velocidad**: Procesamiento muy rápido
3. **Transparencia**: Fácil entender por qué acepta/rechaza
4. **Control**: Total control sobre qué se acepta
5. **Recursos**: Requiere mínimos recursos computacionales

5.2. Ventajas del NLP Moderno

1. **Flexibilidad**: Maneja variaciones y errores
2. **Escalabilidad**: Vocabulario extenso
3. **Inteligencia**: Resuelve ambigüedades usando contexto
4. **Aprendizaje**: Mejora con más datos
5. **Generalización**: Funciona con texto no visto antes

5.3. Limitaciones del Parser Formal

1. **Vocabulario limitado**: Solo palabras predefinidas
2. **Rigidez**: No tolera variaciones

3. **Mantenimiento**: Requiere actualizar gramática manualmente

4. **Ambigüedad**: No puede resolver ambigüedades

5. **Contexto**: No usa información contextual

5.4. Limitaciones del NLP Moderno

1. **Recursos**: Requiere modelos grandes

2. **Velocidad**: Más lento que parser formal

3. **Transparencia**: Difícil entender decisiones

4. **Errores**: Puede cometer errores sutiles

5. **Dependencias**: Requiere bibliotecas externas

6. Conclusiones

6.1. Hallazgos Principales

1. **Los parsers formales son excelentes para dominios específicos** donde se requiere validación estricta y se conoce la estructura exacta.

2. **El NLP moderno es superior para procesamiento general** donde se necesita flexibilidad y manejo de ambigüedades.

3. **La combinación de ambos enfoques** puede ser poderosa: parser formal para validación rápida, NLP para análisis profundo.

4. **La gramática libre de contexto tiene limitaciones** para lenguaje natural completo, pero es adecuada para subconjuntos controlados.

6.2. Aplicaciones Prácticas

• **Parser Formal**: Validación de formularios, comandos estructurados, DSLs

• **NLP Moderno**: Análisis de texto libre, chatbots, traducción, resumen

6.3. Lecciones Aprendidas

1. La implementación de un parser descendente recursivo es más flexible que LL(1) pero menos eficiente.

2. El diseño de la gramática es crítico: debe balancear expresividad y simplicidad.

3. El vocabulario limitado es una restricción importante pero necesaria para mantener la gramática manejable.

4. La comparación con herramientas modernas muestra claramente las ventajas y desventajas de cada enfoque.

7. Referencias

• Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). *Compilers: Principles, techniques, and tools* (2nd ed.). Pearson Addison-Wesley.

• Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed.). Prentice Hall.

• spaCy Documentation: https://spacy.io/