Universidad Nacional de Colombia

Compiladores 2022 - II

Proyecto 1

4 de octubre de 2022

Lara Camacho, Juan David.
Sanchez Orjuela, Juan Carlos.
Sanchez León, Stiven Leonardo.

. . . . . .

Parte A

Desarrollamos una herramienta de generación de escáner definido como en [2] mediante el uso de Python 3.10 que podemos encontrar en Github (link).

Parte B

Desarrollamos un parser para el lenguaje JSON para ello empezamos especificamos los tokens para el lenguaje:

Informalmente

Describimos primero informalmente los tokens para este lenguaje basados enla gramáticas deifinida en [1] pero sin usar todos por lo extenso que pueda llegar a ser la implementación.

1. Nombre: corresponde a la cadena que define los atributos, objetos o arreglos, esta cadena debe empezar con una letra y luego puede tener letras y números.

2. Cadena: corresponde a los strings de texto, se debe escribir solo con letras y números y puede en principio ser vacía.

3. Entero: corresponde a una secuencia de dígitos numéricos y debe haber almenos uno.

4. Racional: corresponde a una escuencia de dígitos donde opcionalmente tiene una parte decimal.

5. Booleano: puede tomar los valores de verdad o falsedad respectivos del lenguaje (true o false).

6. Null: corresponde a dato nulo que se escribe null en este caso.

7. Asignación: corresponde a el caracter con el cual se definen los atributos, en este caso ":".

8. Separación: corresponde al caracter de

separación de atributos el cual en este caso es ",".

9. Comillas: "
10. I. obj: `{`
11. F. obj: `}`
12. I. arr: `[`
13. F. arr: `]`

Formalmente

Basado en las anteriores definiciones procedemos a definirlas formalmente, es decir, mediante la expresión regular que lo caracteriza.

1. Nombre: `[a-zA-Z][a-zA-Z0-9]*`
2. Cadena: `[a-zA-Z0-9]*`
3. Entero: `[0-9][0-9]*`
4. Racional: `[+-]?([0-9]*[.])?[0-9]+`
5. Booleano: `true`|`false`
6. Null: `null`
7. Asignación: `:`
8. Separación: `,`
9. Comillas: `"`
10. I. obj: `{`
11. F. obj: `}`
12. I. arr: `[`
13. F. arr: `]`

Ahora procedemos a construir un parser con lo anterior, para ello definimos la siguiente gramática libre de contexto (CFG):
Los simbolos terminales corresponden justamente a los tokens definidos anteriormente y los no terminales P y E para programa y expresión respectivamente. Las reglas de asignación corresponden a las siguientes:

Gramática para JSON

1.	P	→	E
2.	E	→	E
3.	E	→	E, E
4.	E	→	E : E
5.	E	→	[E]
6.	E	→	"Nombre":
7.	E	→	: Entero
8.	E	→	: Racional
9.	E	→	: Booleano
10.	E	→	: Cadena
11	E	→	: Null

Esto nos permite derivar las correctas oraciones en el lenguaje de **JSON**.

Adicional vemos en la figura – la representación gráfica del autómata (grafo) construido usando lo desarrollado en la parte A.

Parte C

- Explique brevemente en que consiste la tokenización por subpalabras (subwords).
  La tokenizacion por subpalabras tiene como idea fundamental la división de palabras en bloques que por si mismos tengan un significado y de los cuales se puedan generar nuevos significados al ser concatenados con otros bloques. Por ejemplo, en inglés *boys* se puede descomponer como *boy*, niño, y *s*, el plural. Esto dota a cada modelo que trabaja de esta manera de una gran versatilidad al reducir mucho de su vocabulario a subpalabras con altos significado.

- Explique brevemente en que consiste el Byte-Pair-Encoding.
  El algoritmo Byte-Pair-Encoding(BPE) consiste en repetidas iteraciones, encontrasr la cadena binaria que más se repite en un *corpus*, un texto origen, y representar los textos de este con un simbolo auxiliar para esta cadena. Esto permite reducir la cantidad de tokens necesarios para representar un texto.

- Explique brevemente en que consiste el algoritmo unigram.
  El algoritmo unigram busca predecir que palabra sigue después de una frase basada en la probabilidad de que una palabra aparezca en un *corps*. Es un caso particular de un n-grama, que usa la probabilidad condicionada de que se dé una palabra dado las últimas n palabras de una frase.

- Explique la diferencia entre estas estrategias y los parsers basados en lenguaje natural.
  Los parser basado en lenguaje natural tienen como fin el dar sentido a textos en un lenguaje basado en reglas gramaticales, esto les permite dar significados sensatos a las oraciones. Pero los métodos anteriormente vistos intentan este mismo fin haciendo uso de procesos estadísticos, esto no permite ver en muchas ocasiones el significado real de una oración al carecer de la información que una gramática puede ofrecer.

Referencias

[1] *Mckeeman form.*

[2] A. V. Aho and A. V. Aho, eds., *Compilers: principles, techniques, & tools*, Pearson/Addison Wesley, Boston, 2nd ed ed., 2007. OCLC: ocm70775643.