Arellano Granados Angel Mariano
218123444

Actividad 6 Aplicaciones De Gramáticas Y Lenguajes Formales

1. Lenguajes de Programación

Los lenguajes de programación se pueden analizar por medio de la sintaxis único de cada lenguaje para posteriormente ser compilada y arrojan el resultado esperado, Las sintaxis de los lenguajes se pueden definir como el formato apropiado de los programas y la semántica como lo que define lo que significan los programas, o en otras palabras los resultados al ejecutarlo.

Una gramática describe en forma natural la estructura jerárquica de la mayoría de las instrucciones de un lenguaje de programación.

Un analizador léxico es aquel que permite un traductor pueda darle un significado a las instrucciones que combinan ciertos caracteres como los identificadores que aun ser varios caracteres se comportan como una unidad conocida como token durante el análisis sintáctico.

Los compiladores tienen un analizador léxico que lee los caracteres del código fuente, donde los agrupa en unidades llamadas lexemas y estos producen en la salida tokens, los tokens consisten en dos componentes el nombre y su valor de atributo, los nombres de los tokens son símbolos abstractos que se les suele llamar terminales ya que aparecen como símbolos terminales en la gramática para un lenguaje de programación, en cambio el valor de atributo es un apuntador a una tabla de símbolos que contiene información adicional del token en cuestión, esta información no es parte de las gramática.

2. Procesamiento de Lenguajes Naturales

Los lenguajes formales también pueden ser un medio por el cual podamos lograr una comunicación natural este el humano y las maquinas, con el tiempo hemos visto el surgimiento de robots, chat bot y autómatas que son capaces de comunicarse con nosotros en nuestro idioma no con la naturalidad que tendríamos con otro humano, pero si simplificando enormemente los procesos de uso a todos aquellos que no comprenden el funcionamiento interno de las maquinas como el público común.

Para esto se tendría que crear una IA que comprenda un lenguaje formal que emule un lenguaje natural como el español o inglés para poderse comunicar con los humanos, para esto se inicia definiendo un léxico que es la lista de palabras permitidas como: sustantivos, pronombres y nombres para denotar

cosas, verbos para denotar sucesos, adjetivos para modificar sustantivos y adverbios para modificar verbos, al ser una cantidad casi infinita de estos y en contante aumento se crean clases abiertas y para las que son pocas serán clases cerradas.

Entonces se deberá crear una gramática para combinar las palabras y crear frases intentando emular los tipos de oraciones que se pretende entienda la maquina como: sentencias, frases nominales, frases verbales, frases preposicionales, oraciones relativas, etc., estas son susceptibles a sobregenera o subgenerar de frases no útiles o carentes de sentido.

Para optimizar la IA se necesita un análisis sintáctico eficiente que se define como el proceso de encontrar un árbol sintáctico para una cadena de entrada dada, así no solo generar oraciones, sino que también será capaz de entenderla analizando cada palabra hasta obtener su propio signo inicial y en propósito de la oración.

Afortunadamente este proceso es aumentable optimizando y aumentando el repertorio de palabras de la IA, para cada vez lograr una comunicación mas natural entre el usuario y la IA.

3. Expresiones Regulares

“Una expresión regular es un modelo o una forma de comparar con una cadena de caracteres. Esta comparación es conocida con el nombre de pattern matching o reconocimiento de patrones, permite identificar las ocurrencias del modelo en los datos tratados. La utilización principal de las expresiones regulares en Perl consiste en la identificación de cadenas de caracteres para la búsqueda modificación y extracción de palabras clave. De esta manera se pueden dividir las expresiones regulares en varios tipos que son: expresiones regulares de sustitución, expresiones regulares de comparación y expresiones regulares de traducción.” (IIB 2023)

Expresiones regulares de comparación

Nos permiten evaluar si un patrón de búsqueda se encuentra en una cadena de caracteres, de modo que mediante este tipo de expresiones regulares obtendremos un valor lógico verdadero o falso según se encuentre el patrón deseado. La sintaxis de este tipo de expresiones regulares es la siguiente:

valor a comparar = ~ patrón de búsqueda

Ejemplo:

```
# imprimimos todas las líneas que contengan "html". Ej.: "htmlexe"

if ($linea =~ /html/) {

    print $linea;

}
```

Expresiones regulares de sustitución

Las expresiones regulares de sustitución permiten cambiar los patrones de búsqueda por caracteres nuevos definidos por el usuario que componen el patrón de sustitución, la sintaxis es la siguiente:

valor a sustituir =~ s/patrón de búsqueda/patrón de sustitución/opciones

Opciones:

Opción	Descripción
e	Evalúa la expresión como una instrucción.
g	Reemplaza todas las ocurrencias.
i	Reemplaza independientemente mayúsculas y minúsculas.
m	Manipula la cadena de caracteres en multilínea.
o	Evalúa la expresión a reemplazar una sola vez.

Ejemplos de su manejo:

```
$var = 'abc123yz';

$var =~ s/d+/$&*2/e;            # $var = 'abc246yz'

$var =~ s/d+/sprintf("%5d",$&)/e; # $var = 'abc   246yz'

$var =~ s/\w/$& x 2/eg;         # $var = 'aabbccc   224466yyzz'
```

Expresiones regulares de traducción

Este tipo de expresiones regulares tienen una manera de trabajar muy parecida a la de las sustituciones. En este caso se trata de comparar uno a uno los caracteres del patrón de búsqueda con los de la cadena de sustitución, de modo que cada vez que se encuentra una ocurrencia que coincide con uno de los caracteres del patrón se intercambia por su correspondiente en la cadena del patrón de sustitución. La sintaxis general de esta expresión regular es la siguiente:

variable =~ tr/patrón de búsqueda/cadena a traducir/opciones

Arellano Granados Angel Mariano
218123444

Opciones:

Opción	Descripción
c	Complementa la lista a reemplazar
d	Borra los caracteres encontrados pero no reemplazados
s	Suprime las repeticiones entre los caracteres reemplazados

Ejemplos:

$var =~ tr/A-Z/a-z/; # transforma mayúsculas a minúsculas

$cnt = $var =~ tr//*/; # cuenta los arteriscos de $sky*

$cnt = $var =~ tr/0-9//; # cuenta y suprime las cifras de $sky

$var =~ tr/a-zA-z//s; # elimina duplicados. bbookk -> bok

$var =~ tr/a-zA-z/ /cs;

 # cambia los caracteres no alfabéticos en espacios y elimina duplicados

4. Aplicación adicional

En informática, los lenguajes formales constituyen la base para la definición de los lenguajes de programación.

En lógica simbólica (también llamada lógica formal), se emplean lenguajes formales para expresar de manera clara y simple las proposiciones y razonamientos, a fin de determinar su validez.

En matemática, los lenguajes formales se emplean para representar relaciones, operaciones y fórmulas.

En lingüística, los lenguajes formales son estudiados en sus aspectos sintácticos, con el fin de comprender las regularidades de las lenguas naturales.

Referencias:

- Compiladores (principios, técnicas y herramientas). Alfred V. Aho et al. Editorial Pearson (Addison Wesley).
- Inteligencia Artificial (un enfoque moderno). Stuart Russell y Peter Norvig. Editorial Pearson (Prentice Hall).

Arellano Granados Angel Mariano
218123444

- IIB. (2023). Perl: Expresiones regulares. Instituto de Investigaciones Biomédicas. https://www2.iib.uam.es/bioinfo/curso/perl/tutoriales/cicei/cap6.htm