Algoritmos Desarrollados

Por Esteban Alvarado Vargas, José Alejandro Ibarra y Jesús Sandoval

Para el desarrollo de *CallCenterLog* fue necesario diseñar tres tipos de algoritmos: analizador léxico, analizador sintáctico y un sistema experto (Figura 1). En esta sección del documento se explica cada uno de los algoritmos que el equipo de desarrollo programó en el lenguaje lógico Prolog, presentando diagramas de flujo y algunos ejemplos del código para una mejor compresión de los mismos.

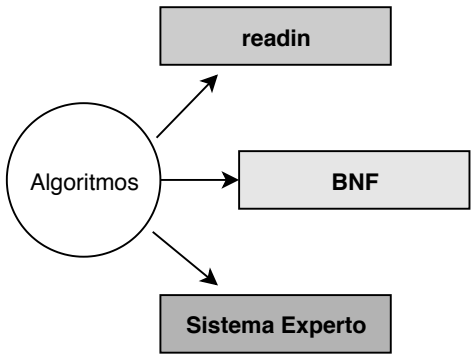


Figura 1. Algoritmos que implementan la solución de CallCenterLog

Etapas

Cuando se recibe una entrada del usuario se sigue una secuencia más o menos general. Esta secuencia está basada en el paper "Logic Programming and Compiler Writing" (David H.Warren, 1980).

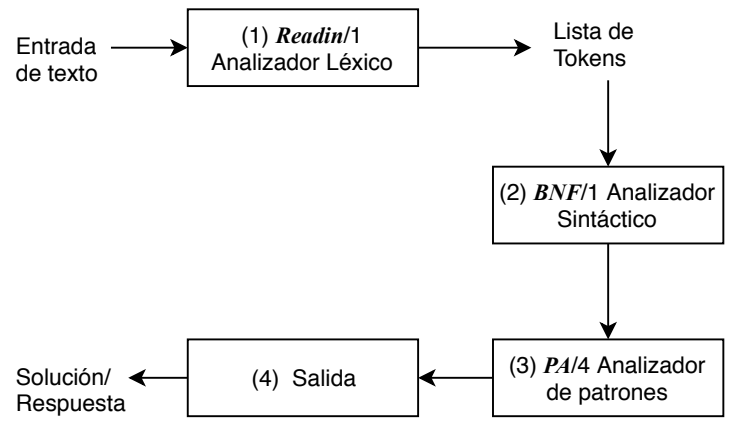


Figura 3. Etapas de Análisis

(1) Readin

Este algoritmo es el analizador léxico del programa, está basado en el código de David H.Warren y Alejandro Pereira como parte del analizador de Lenguaje Natural **Chat-80**. Nuestra implementación le agrega modificaciones que lo adaptan al idioma español.

***Resumen**: Le entra un texto y retorna una lista con Tokens.*

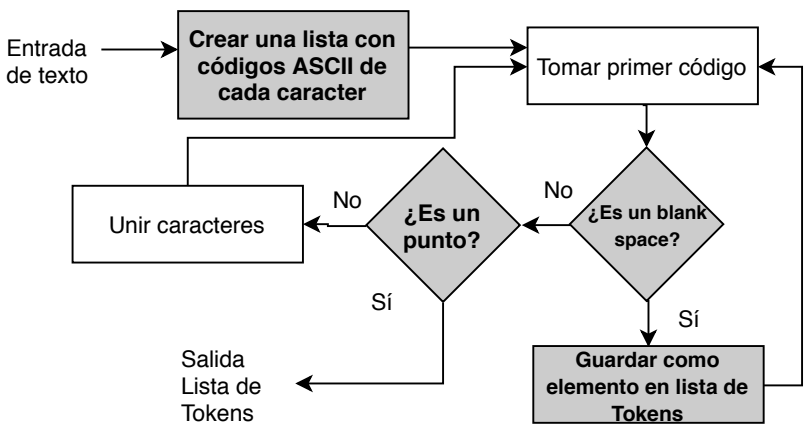


Figura 4. Algoritmo Readin (Analizador Léxico)

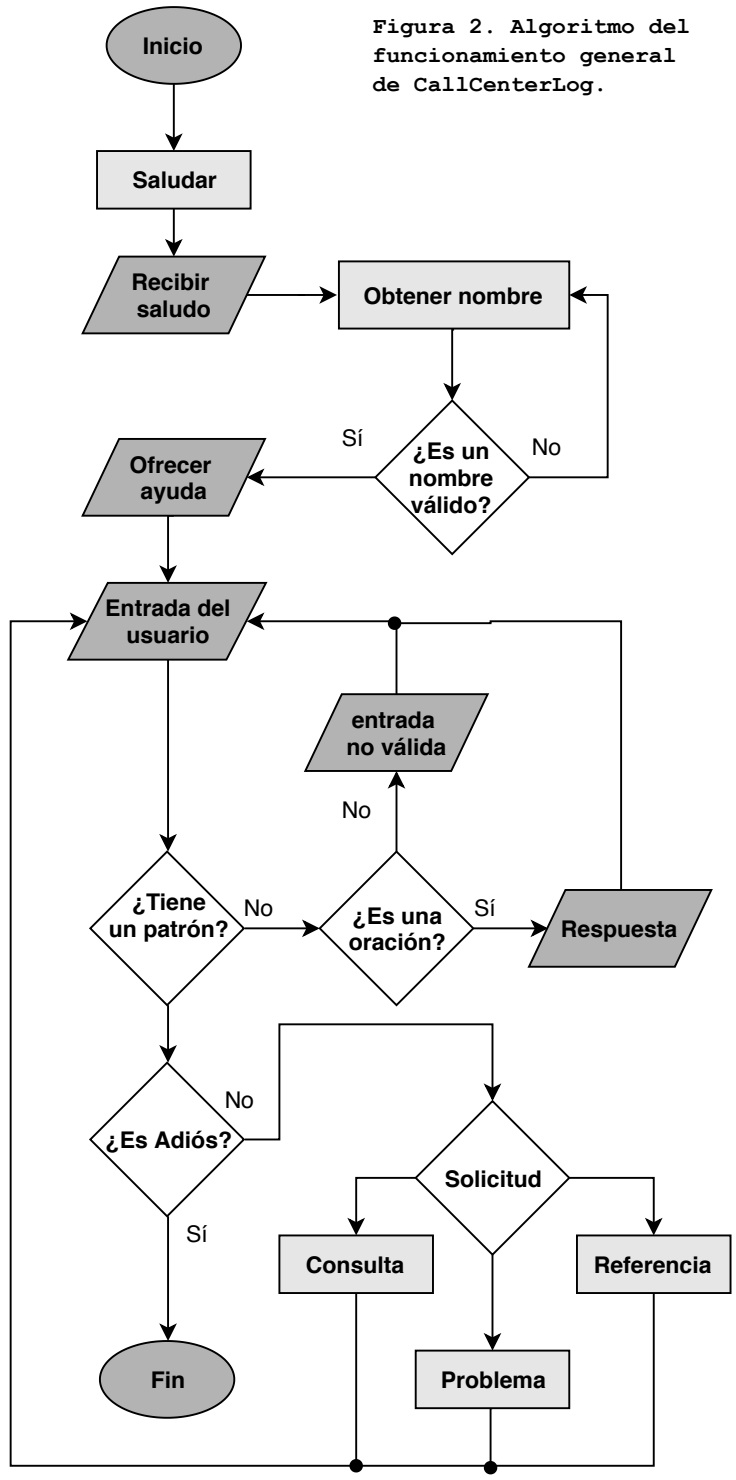


Figura 2. Algoritmo del funcionamiento general de CallCenterLog.

Algoritmo CallCenterLog

El algoritmo de CallCenterLog se puede definir a un alto nivel como la unión de dos partes: *Introducción y Conversación.*

En la introducción el sistema saluda al usuario con un mensaje aleatorio tomado de la base de datos y espera a que el usuario le conteste, para este momento el sistema solo recibirá "lo que sea" que entre y lo aceptará como un saludo. Luego le preguntará al usuario el nombre para almacenarlo en una variable (es útil al final), si el usuario no ingresa su nombre o una oración que lo contenga, el sistema se encargará de volver a pedirlo hasta que introduzca un nombre válido para su base de datos.

Ahora sí, es momento de conversar. En la conversación el bot inicia preguntando al usuario *"¿En qué le puedo ayudar?"* y espera una contestación por parte del usuario. Se procede a realizar el análisis léxico convirtiendo la entrada en lenguaje natural a una lista de **tokens** que el sistema analiza para saber si la entrada es una oración permitida por el analizador sintáctico; si no es una entrada válida lo notifica al usuario e intenta nuevamente. Por el contrario, si es aceptada, entonces busca patrones que le permitan clasificar la entrada en consultas, problemas y referencias. Estos algoritmos se estudian con mayor detalle en sus secciones correspondientes.

***token**: palabra para describir las palabras y símbolos de la entrada en lenguaje natural del usuario..*

(2) Analizador Sintáctico

Para verificar que la entrada del usuario sea una oración válida en el idioma español utilizamos la *gramática libre de contexto [GLC]* (Escrig & Pacheco,nd) en la que sus constituyentes son estructural mente mutuamente independiente, es decir, la estructura de una parte no influye en la estructura de otra.
Formalmente la GLC tiene la siguiente forma:

```
<oracion> -> <sintagma_nominal> <sintagma_verbal>
<sintagma_nominal> → <determinante> <nombre>
<sintagma_verbal> → <verbo> <sintagma_nominal>
<sintagma_verbal> → <verbo>
<determinante> → el
<determinante> → la
<nombre> → hombre
<nombre> → manzana
<verbo> → come
```

Esta notación para describir gramáticas se denomina BNF - *Backus-Naur form* - .
La derivación de una frase puede describirse como un árbol de derivación:

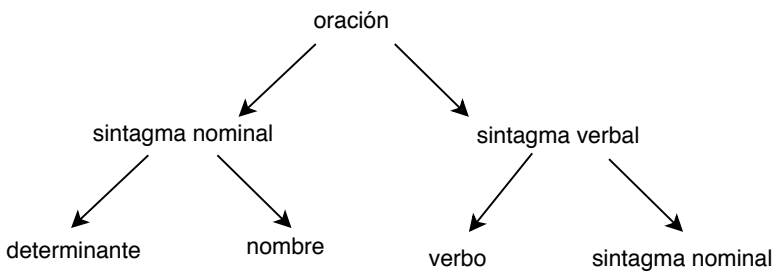


Figura 5. Árbol de derivación

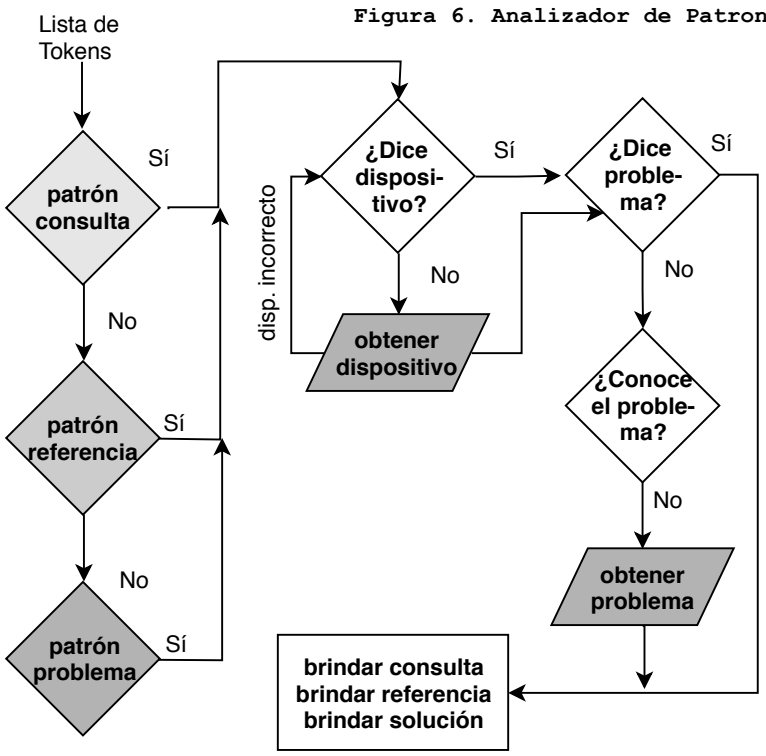
En el código este algoritmo de verificación de oraciones nos permite detectar que las entradas del usuario mantengan cierta coherencia gramatical sintáctica, por lo que es una parte fundamental del algoritmo general, es un filtro que evita que entre al procesador un texto que no sepa manejar. El código se utilza de esta manera:

```
oracion([mi, impresora, esta, mala],_).
true .
```

(3) Analizador de Patrones

En este punto la entrada del usuario ha sido validada y es momento de que el **Sistema Experto** detecte mediante patrones de texto lo que el usuario está solicitando. El analizador se encarga de verificar si la *lista de tokens* contiene algún patrón que permita primero clasificar la solicitud del usuario en consulta, problema o referencia y luego detecte el problema con el que se asocia la solicitud y pueda contestar certera-mente.

Figura 6. Analizador de Patrones



(3.1) Obtener problema

CallCenterLog posee una base de datos de problemas, con su respectiva causa, solución y referencia. El algoritmo se encarga de buscar patrones en la lista de tokens, que encajen con los patrones de la base de datos de causas y problemas generales. Si el usuario conoce el problema, lo ingresa y se encarga de brindar la solución asociada, en el caso de que el usuario no lo conozca, *CallCenterLog* comienza a realizar preguntas de Si o No para ir determinando un diagnóstico del problema. Si el SE no consigue dar con una solución para el problema envía una disculpa al usuario y sale del bucle.

Es importante decir que si se introduce la palabra clave "salir" el bucle se cierra y regresa a la conversación.

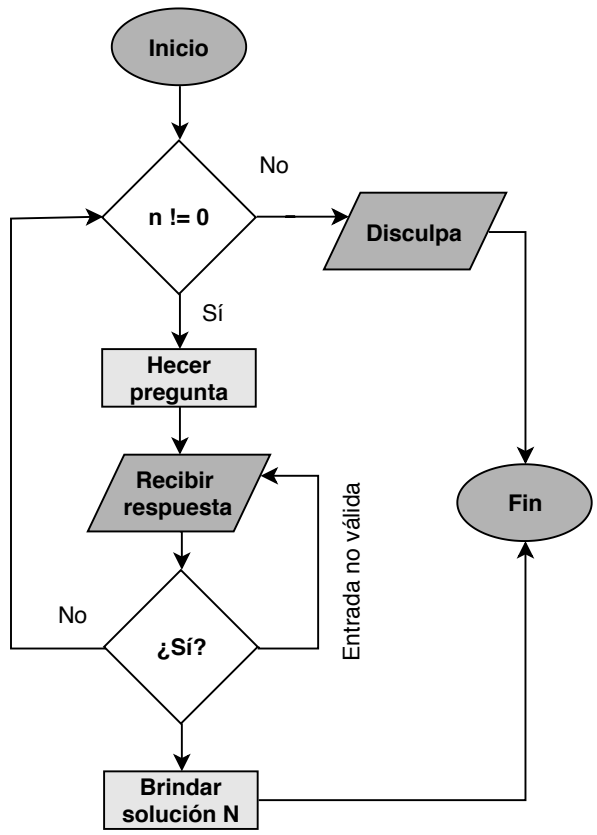


Figura 7. Algoritmo Obtener problema

Referencias

Escrig, M., & Pacheco, J. *El Lenguaje de Programación Prolog* [Ebook] (pp. 71-74).

Warren, D. (1980). Logic programming and compiler writing. *Software: Practice And Experience*, *10*(2), 97-125. doi: 10.1002/spe.4380100203

Warren, D., & Pereira, F. (1980). An Efficient Easily Adaptable System for Interpreting Natural Language Queries (1).