Instituto Tecnológico de Costa Rica

Escuela de Computación

Sistemas operativos

Proyecto l

Estudiantes:
Keslerth Calderón Artavia
Valeria Herrera Rodríguez
Pamela Varela Alfaro

Sede San Carlos
24 de marzo, 2019

1. Introducción

En este proyecto, se implementa la programación secuencial y concurrente tomando las URL´s para clasificarlo por medio del método Bayesiano Ingenuo mediante categorías, y también se incluye la parte distribuida, utilizando el lenguaje de programación C#, la cual soporta las ejecuciones en multicore, además se mostrará las estadísticas y los resultados de las ejecuciones con base en las implementaciones mencionadas.

2. Análisis del problema

En este caso, se debe crear un sistema capaz de clasificar un URLS´s por medio del método Bayesiano Ingenuo. Se deben utilizar dos archivos, el "*dmoz.csv*" que contiene la información con la que se van a clasificar las URL´s, además de que se va a utilizar para crear la base de datos de conocimiento y el archivo "*URL-Clasificacion.csv*" donde están almacenadas las URL´s a clasificar.

Se debe utilizar procesos multicore y programación distribuida para manejar la base de datos de conocimiento y clasificar las URL´s. Se deben crear métodos que permita relacionar una palabra con una categoría, para luego aplicar la fórmula que permitirá establecer una probabilidad a través del Bayesiano para cada URL.

De lo anterior, se tiene que mostrar por medio de un complemento gráfico sobre las probabilidades que tuvieron en cada URL, en fin se tiene ejecutar el proceso secuencial y concurrente de cada uno de los métodos para medir la CPU por medio de estadísticas y visualizarlo por medio de gráficas el uso de los núcleos de la computadora.

3. Solución del problema

Mediante la investigación realizada en el análisis del problema se decidió utilizar el lenguaje de programación C# que tiene capacidades de programación multicore, en el ambiente de desarrollo Visual Studio 2017, además del uso de una base de datos relacional creada en PostgreSQL y la clase Forms para mostrar los datos por medio de interfaz gráfica, como las estadísticas realizadas y el comportamiento de la CPU. Lo anterior, se eligió con el objetivo de realizar de la mejor forma los distintos métodos para la correcta estructuración y creación del proyecto, que se explicará a continuación:

Se creó un método concurrente para crear la base de datos de conocimiento a partir del archivo *"demoz.csv"*. Lo primero que se realizó fue obtener los datos y extraer la información necesaria, a estos datos se les debe quitar palabras, espacios o caracteres innecesarios para la base de datos de conocimientos. Los archivos que se agregan a la base de datos son una palabra y la categoría a la que pertenecen, la representación es la siguiente:

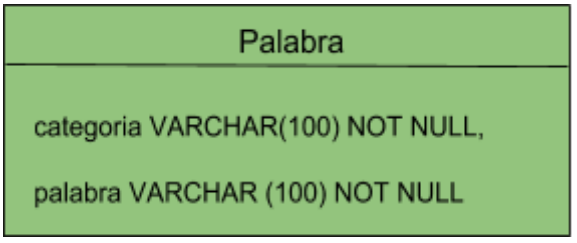

Imagen 1. Entidad de palabra.

Para la clasificación de las URL´s se divide en varios métodos, el primero es obtener los datos del archivo *"URL-Classification.csv"* y los agrega a una lista de objetos global. Para facilitar el proceso y el flujo en la clasificación se crea la siguiente estructura:

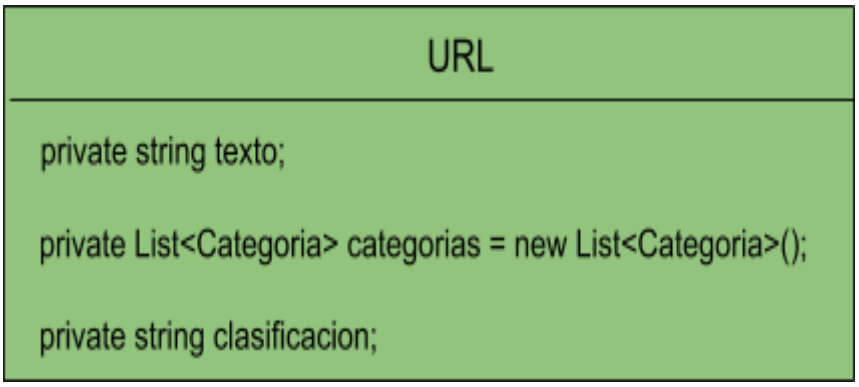

Imagen 2. Clase de la URL.

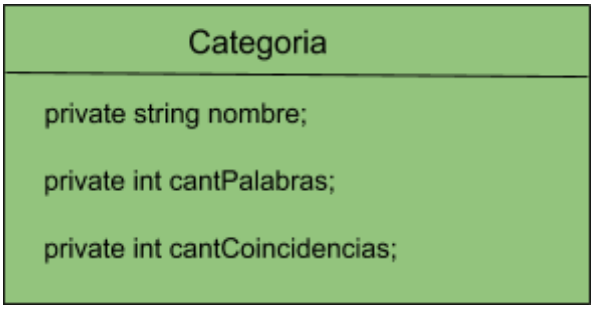

Imagen 3. Clase de la categoría.

Cuando se tienen los datos en la lista de URL´s, se recorre la lista uno por uno para comparar con cada palabra existente en la base de datos de conocimiento, se utiliza el método *.Contains()* de C# para identificar si hay una coincidencia. El objetivo es tener la certeza de cuántas palabras pertenecen a una categoría que están dentro de la URL, entonces si una palabra coincide se debe verificar si la categoría a la que pertenece se encuentra agregada en la lista de objetos de tipo *"categoria"*, en caso de que así sea se debe sumar 1 la cantidad de coincidencias, sino se crea un nuevo objeto categoría con el nombre de la categoría se inicializa en 1.

Cuando se termina de comparar todas las URL´s se empieza método Bayesiano Ingenuo, que necesitan los siguientes datos:

- **`Int64 cantidadTotalBase = pgsql.ObtenerCantidadBase();`**

Es la cantidad total de palabras que se ingresaron a la base de datos de conocimiento de todas las categorías.

- **double totalCategoria = URLs[i].getCategorias()[x].getCantCoincidencias();**

Es el total de palabras que se compararon y coincidieron de una categoría específica.

- **double cantCategoria = Convert.ToDouble(pgsql.ObtenerCantidadCategoria(URLs[i].getCategorias()[x].getNombre()));**

Es el total de palabras que se compararon de una categoría, independientemente que coincidieron o no.

Se necesitó de las tres variables anteriores para realizar la siguiente fórmula:

double cantFinal = (cantCategoria / cantidadTotalBase) * (totalCategoria / cantCategoria);

Con *cantFinal* se obtiene la probabilidad que tiene cada categoría y por último se comparan entre todas y la que tenga el porcentaje más alto es la clasificación final.

Las funciones tanto de clasificación como de Bayes se realizó de dos maneras: concurrente con el método de C# Parallel.Invoke() utilizando 4 hilos y también se realizó de manera secuencial para comparar el comportamiento y la eficiencia en ambos. Además de aplicar sockets para el uso de tres máquinas corriendo el sistema, para hacer la clasificación más eficiente, debido a la gran cantidad de datos que debe procesar.

Para sockets se tiene un máquina que funciona como servidor, que será el encargado de distribuir la tareas y recibir los datos que las máquinas que funcionan como cliente procesen. El servidor identifica la cantidad de clientes que se estén conectado y distribuye la cantidad

de URL que debe clasificar cada cliente, además antes de clasificar se encarga de subir los datos necesarios para la base de conocimiento y las URL's que se van utilizar en el proceso. Le indica a los clientes que funciones son las que deben ejecutar y el momento en que lo debe hacer.

4. Análisis de resultados

A continuación, se muestra una tabla de los resultados para cada requerimiento que se implementó para el proyecto, tomando en cuenta el estado y las respectivas observaciones.

Requerimiento	Estado	Observaciones
Método de concurrencia para la llenar la base de conocimiento de cada palabra con su categoría.	100%	
Método secuencial para la clasificación de las URLs con la fórmula de Bayesiano Ingenuo.	100%	
Método concurrente para la clasificación de las URLs con la fórmula de Bayesiano Ingenuo.	100%	
Implementación de la programación distributiva.	100%	
Mostrar estadísticas y resultados de la ejecución multicore.	100%	
Mostrar estadísticas y resultados de la ejecución distributiva.	100%	

5. Conclusiones

La implementación de la concurrencia fue muy sencilla gracias a que el lenguaje C# deja implementar este concepto de forma sencilla, a pesar de que el utilizar la concurrencia es algo difícil de hacer y comprender. Se pudo ver que hubo una mejora considerable a la hora de ejecutar secuencial y concurrente.

En la parte de la programación distribuida se utiliza un concepto llamado sockets el cual lo que hace es enviar constantemente mensajes entre sí para tener conocimiento de lo que quiere realizar uno y el otro. Con este concepto se pudo ver que el asignar una cantidad a cada computadora ayudaría a mejorar la velocidad a la que termina un proceso una computadora, pero al hacerlo de esta forma se puede ver como varias computadoras realizan un mismo proceso para terminarlo en el menor tiempo posible.

Sin embargo de lo difícil que fue aprender cómo implementar el concepto de distribución al proyecto se pudo realizar un tipo de distribución y entender el funcionamiento de la misma, y poder apreciar la cantidad de tiempo que ahorra una computadora al utilizar casi al 100% su procesador es fascinante y desarrollar proyectos nuevos con retos como estos generan estrés pero a la vez se aprende de herramientas nuevas para nuestro desarrollo como futuros trabajadores en el área del software.

6. Referencias

- Rpetrusha et al. (s.f.). How to: Use Parallel.Invoke to Execute Parallel Operations. Disponible en: https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/how-to-use-parallel-invoke-to-execute-parallel-operations

- Statsoft. (s.f.). Naive Bayes Classifier Disponible en: http://www.statsoft.com/textbook/naive-bayes-classifier