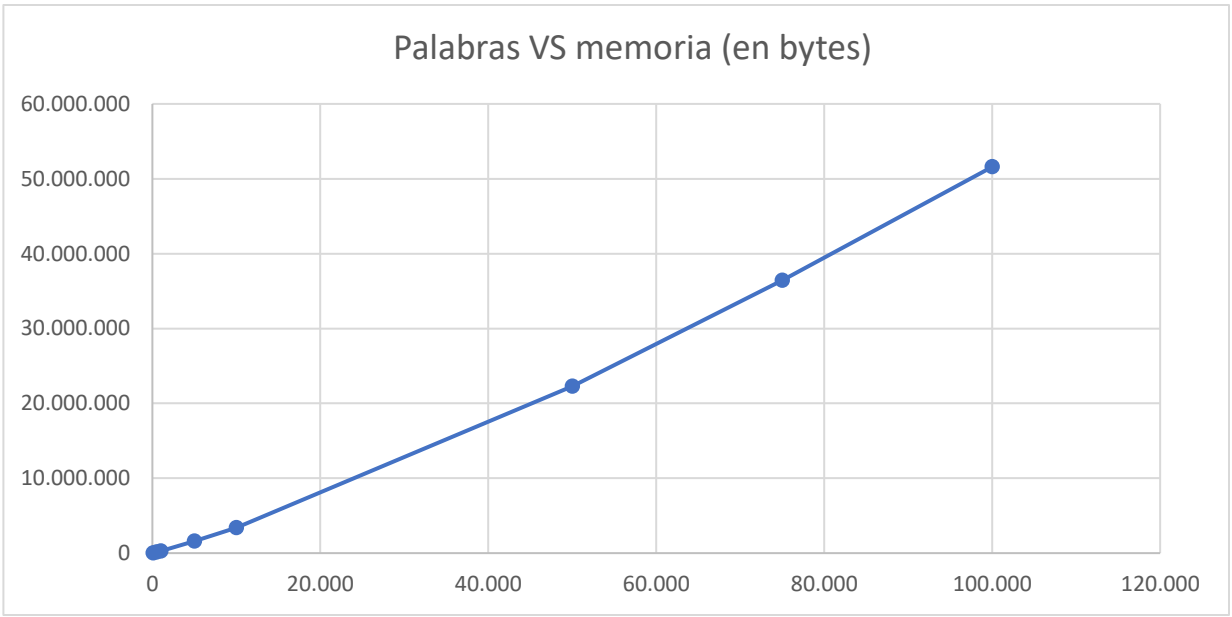
Proyecto 1: Texto Predictivo

Estructuras de Datos, Primer Semestre 2020

Prof. Diego Seco

Ayudantes: Catalina Pezo y Alexis Espinoza

Integrantes: Cerda Saavedra Felipe & Schultz Solano Vicente



Nota: Se usaron palabras de 2 a 10 letras. La cantidad de letras por palabra fue en acenso en función a la cantidad de palabras. Es decir, a más palabras más letras por palabra. Dicho esto, en general, el diccionario usado tiene mas palabras con letras entre 4-6 que entre 2-3, 7-10+.

Descripción de la solución.

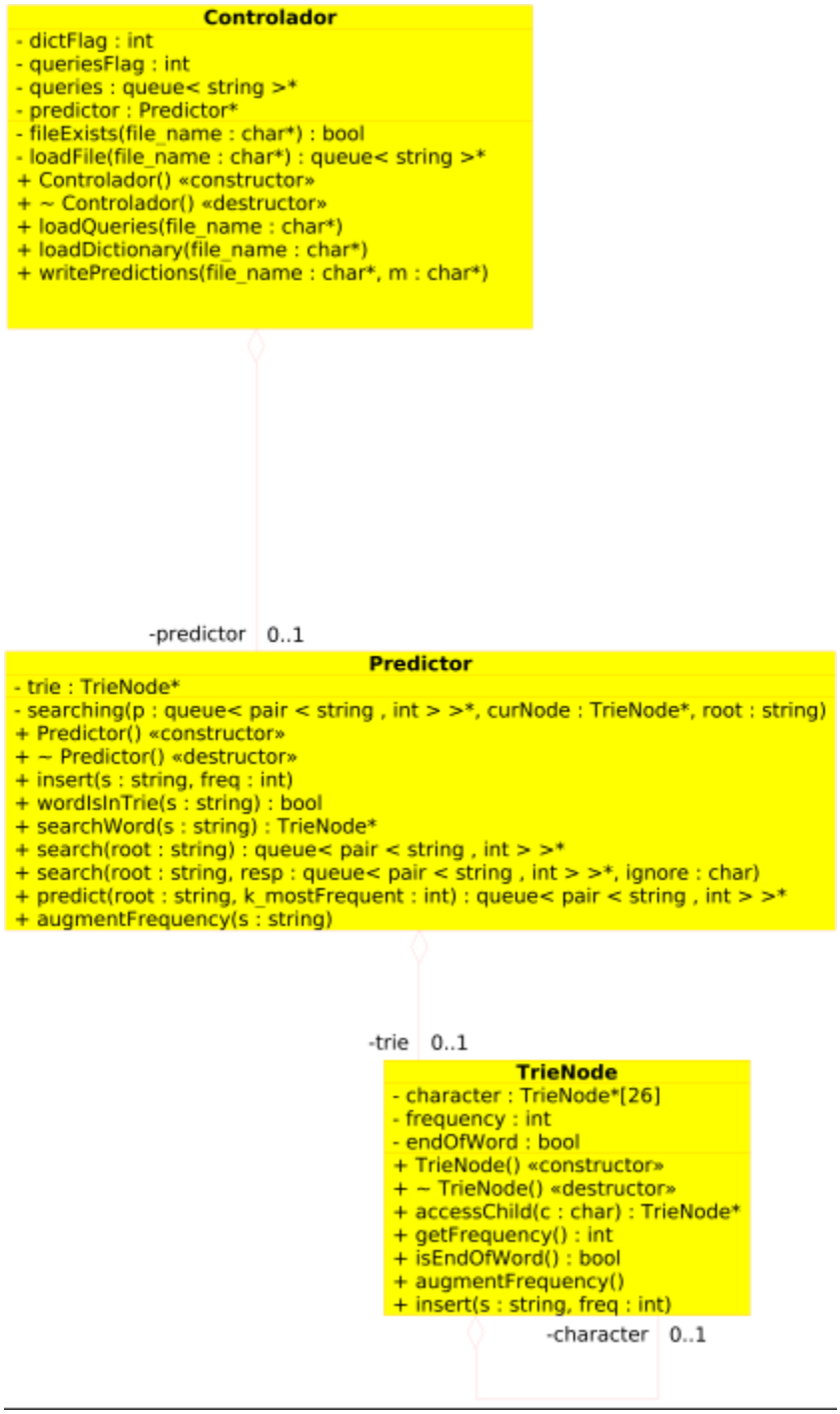
Estructura del proyecto

La solución propuesta consta de tres clases, a saber, Controlador, Predictor, TrieNode. La clase Controlador se encarga de procesar los archivos de diccionario y consultas y de generar el archivo con respuestas. Contiene en su interior una instancia de la clase Predictor.

Por su parte, la clase Predictor contiene la instancia de la clase TrieNode, trie, y se encarga de insertar las palabras que recibe desde la clase Controlador. A partir de ellas lee las palabras que hay en "trie" y genera las predicciones.

Finalmente, la clase TrieNode, es la implementación de la estructura de datos "Trie" y almacena las palabras y maneja el acceso a estas.

A continuación, adjuntamos el diagrama de clases.



1. Leer palabras desde archivo diccionario y archivo de consultas (Controlador)

Para leer las palabras desde el archivo diccionario, utilizamos el método ***loadDictionary***. Dicho método verifica que el archivo exista en disco y luego lee línea por línea limpiando comentarios (aquello marcado con "%"). Posteriormente envía las palabras a Predictor para que este las inserte en "trie". De manera análoga, el método ***loadQueries*** lo hace para el archivo de consultas con la diferencia que, en vez de entregárselas a Predictor, las guarda como un queue de String llamada queries.

2. Insertar palabras

Una vez que Predictor recibe las palabras desde Controlador, este las pasa a TrieNode para su inserción. La clase TrieNode es la que se encarga de insertar palabras, en consecuencia, predictor no puede modificarlas directamente y en caso de necesitar modificarlas solo puede hacerlo a través de los métodos que TrieNode proporciona: ***insert*** y ***augmentFrequency***.

3. Para buscar palabras y predicción

Una vez que tenemos las palabras dentro de trie y cargadas en memoria la lista de consultas realizamos la predicción. El método ***predict*** de la clase predictor recibe las palabras a predecir y un entero indicando que debe devolver las k-mas-frecuentes. El funcionamiento de ***predict*** a grandes rasgos consiste en buscar recursivamente las k-palabras-mas-frecuentes. Si no encuentra suficientes palabras, realiza la búsqueda quitando la letra final e ignorando el camino que siguió la vez pasada. Realiza este último proceso todas las veces que sea necesario.

4. Escritura de archivo con predicciones.

El método ***predict*** de Predictor devuelve la lista de predicciones al metodo ***writePredictions*** y este se encarga de escribirlas en disco.

Detalles de la implementación.

Para comprender completamente el funcionamiento de la solución es necesario precisar la implementación de trie, el método de inserción y, en términos generales, la forma de buscar dentro del trie.

Descripción de Trie

La clase TrieNode representa un nodo de Trie con referencia a otros nodos. Contiene los siguientes miembros, ***character*** que corresponde a un ***array*** de 26 TrieNode, uno por cada letra del alfabeto. También contiene ***frequency*** y ***endOfWord*** para guardar la frecuencia de las palabras y para señalar el fin de una palabra respectivamente.

Descripción de método de inserción

Inserta las palabras en forma de árbol. Por ejemplo, partiendo desde la raíz, todas las palabras que comienzan con 'a' siguen el camino hacia ***character[0]*** y desde ahí, letra por letra, hacia el siguiente nodo. Para clarificar, si insertamos la palabra "pala", desde la raíz insertara:

root.character[16]>>p.character[0]>>a.character[11]>>l.character[0] marcará el ultimo nodo como fin de palabra, ***isEndOfWord***, y le asignará una frecuencia, ***frequency***. Si luego insertamos "paladin", recorrerá desde la raíz el camino de la palabra anterior y solo insertará nodos para el Substring 'din'.

Método de búsqueda de palabras dentro del trie

Para cualquier palabra buscada se entra al nodo raíz y, desde ahí, se mueve al nodo que corresponde a la 1era letra de la palabra. Desde ahí nos movemos iterativamente al nodo que corresponde a las letras siguientes. Se sigue iterando hasta llegar al nodo donde se encuentra la ultima letra de la palabra marcado como endOfWord. Todos los métodos de búsquedas siguen la misma lógica.