Sistema de texto predictivo e indexación de nombres por sonido

Juan Guillermo Lalinde, José Luis Pareja, Alberto Restrepo,
Mauricio Toro y Édison Valencia

Proyecto para Estructuras de datos y algoritmos 1. Programa de Ingeniería de Sistemas.
Tamaño del grupo: Grupos de 2 personas

1. El problema

Uno de los grandes retos que ha perseguido a la Ingeniería de Sistemas, casi desde sus origenes, es desarrollar aplicaciones que presenten comportamientos inteligentes. Los problemas de indexación de nombres por sonido, autocompletar textos, corrección de ortografía y algoritmos de separación por sílabas clasifican en esta categoría y son de gran interés en este proyecto.

El objetivo principal de este de este proyecto es hacer el análisis de los algoritmos y estructuras de datos, y, finalmente, implementar una solución eficiente al siguiente problema: Se necesitan almacenar, en una estructura de datos eficiente, un conjunto grande de palabras en Inglés y permitir las siguientes consultas: (1) dado una palabra, retornar las palabras con una fonética ortoepía inglesa similar, (2) dado una palabra, retornar las palabras que inician con esa palabra, ordenadas de mayor a menor según el número de consultas de comparación fonética que éstas han tenido.

2. Ejemplos

2.1. Fonética ortoepía

Como un ejemplo, si uno busca palabras que "suenen" similar a "mouse", el sistema deberá entregarle palabras como:

- house
- trouse
- blouse
- grouse
- rouse
- ...entre otras.

Las palabras que se buscan en comparación fonética siempre van en orden alfabético.

2.2. Palabras que inician con una palabra

Como otro ejemplo, cuando el sistema no ha sido utilizado y uno busca palabras que empiecen con la cadena "hollow", en el diccionario que está en el anexo, el sistema deberá entregarle las siguientes palabras:

- hollow
- holloware
- hollowed
- hollower
- hollowest
- hollowfaced
- hollowfoot
- hollowhearted
- hollowheartedness
- hollowing
- hollowly
- hollowness
- hollowroot
- hollows
- hollowware

No obstante, a medida que se hagan buscas de comparación fonética, las palabras que entrega autocompletar, deberán ser ordenadas según el número de búsquedas que éstas han tenido en comparación fonética. Por esa razón, en un futuro, el sistema podría entregar una salida como esta:

- hollowly
- hollowed
- hollow
- holloware
- hollower
- hollowest
- hollowfaced
- hollowfoot
- hollowhearted
- hollowheartedness
- hollowing
- hollowness
- hollowroot
- hollows
- hollowware

En la que "hollowly" y "hollowed" aparecen antes que las demás porque éstas han sido consultadas más veces en las consultas de comparación fonética.

3. Las entregas del proyecto

1. **Documentación de problemas similares.** Informe técnico usando la plantilla ACM. Escribirlo en Inglés preferiblemente.

   a) EN EL INFORME PDF. Explicar al menos 4 soluciones a problemas similares que se encuentren **documentados** en libros, artículos científicos o páginas web. Referenciar las fuentes usando el formato para referencias fuentes que plantea la ACM `http://www.cs.ucy.ac.cy/~chryssis/specs/ACM-refguide.pdf`.

2. **Implementación de un primer prototipo.** En el prototipo usted debe definir sus propias estructuras de datos para representar el conjunto de palabras e implementarlos algoritmos fundamentales que necesitará para las consultas. Además, actualice su informe técnico usando la plantilla ACM, sin borrar lo anterior que tenía en el informe.

   a) EN EL INFORME EN FORMATO PDF:

      1) Explicar la estructura de datos sobre la que trabaja el algoritmo. Ejemplos de estas estructuras de datos son tries, árboles binarios, árboles binarios de búsqueda, listas, arreglos y tablas de Hash.
      2) Seleccionar unos algoritmos para solucionar el problema eficientemente y explicar por qué los seleccionaron. Deben defender por qué eligieron esos algoritmos.

3) Calcular la complejidad de las operaciones para el peor de los casos del algoritmo en términos de 3 variables: longitud máxima de una palabra en el conjunto de palabras (n), longitud de la palabra que recibe la consulta (m) y número de palabras del diccionario (p).
4) Calcular (i) el tiempo de ejecución y (ii) la memoria usada, para las dos operaciones requeridas, para varios ejemplos.

b) EN EL CÓDIGO COMPRIMIDO EN .ZIP:

1) Implementar un prototipo de la solución al problema para un conjunto de palabras en Inglés. Probarla con el que está en el anexo.
2) Recibir el tipo de consulta y la palabra para la consulta por la entrada estandar, por ejemplo `programa -autocompletar hel` o `programa -similitudfonetica hello`.
3) Incluir la documentación HTML del código

c) EN UN DOCUMENTO EN FORMATO PDF ANEXO

1) La documentación de progreso gradual y trabajo en equipo (si aplica).

3. (100 %) **Implementación de un sistema de indexación de nombres por sonido y texto predictivo.** Informe técnico final usando la plantilla de la ACM en formato PDF, código comprimido en .ZIP y anexo en formato PDF.

a) (10 %) **Tabla de comparación entre alternativas de solución.** Como mínimo dos algoritmos que solucionan el problema, de los indicadores de eficiencia (tiempo de respuesta, número de operaciones que debe efectuar el algoritmo y recursos físicos como la memoria). **Para evaluar esto**, se valida los argumentos orales que los estudiantes den en el momento de la presentación del trabajo al profesor como parte de la nota en el rango según el número de algorítmos implementado.

b) (20 %) **Implementar una solución al problema.** Implementar la estructura de datos para el diccionario de palabras y los algoritmos para las consultas, y recibir los parámetros para las consultas por la entrada estandar. **Para evaluar esto**, el programa debe recibir por la entrada estandar los parámetros para las consultas y el tipo de consulta.

c) (15 %) **Calidad del código.** En este criterio se évalúa que el código esté correctamente identado, el nombre de las variables sea claro, que tenga comentarios descriptivos y exista una documentación en formato HTML, y que tenga en cuenta el acoplamiento y la cohesión. **Para evaluar esto**, deben documentar cada clase y método y generar una documentación HTML.

d) (40 %) **Selección del algoritmo.** En este criterio se evalúa cómo determinaron que la solución propuesta es válida para el problema, es decir, cómo determinaron que permitía tener un tiempo de ejecución menor al tiempo especificado por el profesor. Para explicar la selección del algoritmo, pueden apoyarse en libros, artículos científicos, páginas web y relatos anecdóticos de expertos. **Para evaluar esto**, deben explicar por qué seleccionaron el algoritmo.

e) (5 %) **Informe final.** En el informe expliquen el análisis de complejidad del algoritmo utilizado, cuáles son las estructuras de datos utilizadas, por qué fueron seleccionadas, qué problemas relacionados encontraron, cuáles problemas tuvieron en el desarrollo y cómo se utiliza la aplicación (4 páginas máximo). También incluir (1) el tiempo de ejecución promedio, (2) la memoria usada por el programa, para los ejemplos dados en el anexo. Incluir imágenes que expliquen el algoritmo y la estructuras de datos, y referencias. Incluir el nombre del profesor como un autor, de último. Deben incluir referencias de todos los problemas relacionados que estudiaron y fuentes que consultaron. **Para evaluar esto**, deben entregar el informe en formato PDF.

f) (5 %) **Progreso gradual.** En este criterio se evalúa que haya habido un progreso gradual dentro del semestre. Se recomienda subir semanalmente los avances en la implementación y el informe a un repositorio en la nube usando git o svn, y hacer las entregas en las fechas indicadas en Interactiva. Tendrá un descuento del 5 % de la nota de la práctica quienes hayan entregado tarde la primera o la segunda entrega entrega, y un descuento del 10 % quienes no incluyan en la implementación final y en el informe final lo solicitado en la primera y segunda entrega. **Para evaluar esto**, se revisan las entregas parciales en Interactiva.

g) (5 %) **Presentación de diapositivas.** Mostrar el análisis de complejidad del algoritmo utilizado, cuáles son las estructuras de datos utilizadas, por qué fueron seleccionadas, qué problemas relacionados encontraron, cuáles problemas tuvieron en el desarrollo y cómo se utiliza la aplicación (6 diapositivas máximo). **Para evaluar esto**, entregan las diapositivas en formato PDF con el formato propuesto. Exponen en 10 minutos.

h) (10 % EXTRA) **Trabajo en equipo.** En este criterio se evalúa que hayan trabajado en equipo. Se recomienda documentar con actas todas las reuniones que haga el grupo de trabajo. **Para evaluar esto**, entregar copia de todas las actas de reunión y el reporte de git o svn con los cambios en el código y quién los hizo. Todo esto va en un documento anexo en formato PDF.

i) (10 % EXTRA) **Reporte en arXiv.** El reporte es aceptado en el repositorio mundial de reportes de ingeniería de sistemas, arXiv (`http://arxiv.org/`). **Para evaluar esto**, incluyan en el reporte en formato PDF y en las diapositivas un vínculo al URL donde quedó el reporte.

4. Prácticas para el desarrollo del proyecto

- Una práctica ágil de desarrollo de Software, incentivada por la comunidad *Xtreme programming*, es la *programación en parejas*. Esta práctica consiste en que dos programadores trabajen al mismo tiempo en un solo computador. Uno de los desarrolladores, el *conductor*, escribe el código; mientras que el otro, el *revisor*, analiza cada línea de código que se digita y señala posibles errores u optimizaciones. Después de cierto tiempo, los desarrolladores intercambian roles. En este proyecto, ustedes debe utilizar esta práctica de desarrollo ágil. **El conductor es responsable de subir a la nube, con su cuenta de git o svn, las actualizaciones que se hicieron mientras él era conductor.**

5. Algunos problemas relacionados

- `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`
- `https://en.wikipedia.org/wiki/Autocomplete`
- `https://en.wikipedia.org/wiki/Predictive_text`
- `https://en.wikipedia.org/wiki/Soundex`
- `https://en.wikipedia.org/wiki/Match_rating_approach`
- `https://en.wikipedia.org/wiki/Trie`
- `https://en.wikipedia.org/wiki/Hash_trie`
- `https://en.wikipedia.org/wiki/Deterministic_acyclic_finite_state_automaton`

6. Rúbrica de calificación

Criterio	**Excelente**	**Bueno**	**Malo**
Comparación de alternativas	Más de 4	2 o 3	Menos de 2
Funcionalidad	Autocompletar y comparación fonética	Uno de los dos	Ninguno de los dos funciona
Calidad de Código	Doc html, acoplamiento, cohesión, identación, nombre de variables	3 de 5 elementos	Menos de 3
Selección de la estructura de datos	Totalmente objetivo	Hay argumentos objetivos y otros subjetivos	Todo es subjetivo o no sabe / no responde.
Informe final	Tiempo de ejecución, memoria usada, explican algoritmos, estructura de datos, referencias, argumenta la selección de la ED	3 de 6 elementos	Menos de 3
Progreso gradual	Hicieron la entrada 1 y 2	Hicieron al menos 1	Ni una
Diapositivas	Tiempo de ejecución, memoria usada, explican algoritmos, estructura de datos, referencias, argumenta la selección de la ED	3 de 6 elementos	Ni uno
Trabajo en equipo	Actas de reunión y reporte en git o svn con los cambios que hizo cada uno	Uno de los 2 elementos	Ninguno
Reporte en ArXiv	Aceptado		No aceptado