Universidad de Granada
Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación
Grado en Ingeniería Informática
Modelos de Computación - A2
Javier Rodríguez Rodríguez

MedidorMetrica.l: proyecto Flex para conteo de sílabas métricas

Flex++ es una herramienta de análisis lexicográfico capaz de, mediante un fichero plantilla, generar código C++ capaz de recibir como entrada un archivo de texto, procesarlo y realizar acciones en función de la estructura del texto recibido, determinada mediante expresiones regulares. Ante el requisito de utilizar la herramienta para resolver un problema, se optó por buscar un problema donde el análisis lexicográfico fuera de más peso que la implementación computacional, y por tanto, uno de naturaleza lingüística: **contabilizar la métrica de todos los versos de un poema en español dado.**

Definición del problema

Si bien el universo de la poesía es uno con múltiples modalidades, el conteo de sílabas métricas en el idioma español está determinado a unas reglas específicas. El problema de fondo consiste en la separación en sílabas de las palabras, y luego contabilizar sílabas teniendo en cuenta las reglas puntuales de la poesía. Por tanto, podemos desglosar la implementación de la solución en los siguientes pasos:

1. **Separación de palabra en sílabas:**

El proceso de separación en sílabas mediante expresiones regulares está fundamentado en la fórmula silábica: la descomposición de la sílaba en ataque, núcleo y coda, donde cada parte tiene una estructura específica.

$$[_{At}\ (C_1)(C_2)][_{Nuc}\ V][_{Cod}\ (C_3)(C_4)] \begin{cases} C_1 = \text{cualquier consonante} & \text{si } C_2 = \emptyset \\ C_1 = /g, k;\ t;\ b, p;\ f/ & \text{si } C_2 = /l/ \\ C_1 = /g, k;\ d, t;\ b, p;\ f/ & \text{si } C_2 = /r/ \\ V = \text{vocal, diptongo, ...} & \\ C_3 = \text{consonante alveolar(final de palabra)} & \\ C_4 = /s/ \text{ (en interior de palabra)} & \end{cases}$$

Fórmula silábica. Wikipedia, 2019

Para esta construcción se crean inicialmente reglas para enlistar consonantes y vocales. Pero como es necesario determinar diptongos, la regla de vocales se separa en 2, vocales abiertas y vocales cerradas. Luego se crean reglas para distinguir ataque y núcleo. La coda, en cambio cuenta con un problema: ha de conocer qué carácter sigue a la consonante detectada para saber si ésta pertenece a la sílaba en construcción o al ataque de la siguiente. Por suerte Flex ofrece el metacaracter para expresiones regulares /, que contempla en el emparejamiento con expresiones regulares el contenido que lo sigue sin consumirlo directamente. Sin embargo, este metacaracter es de uso riesgoso y no se puede

encapsular, por lo que habrán de definirse directamente las múltiples reglas que contemplen cada una de las posibles codas, aunque la acción a efectuar sea la misma.

Los tokens extraídos, las sílabas, son apilados en un vector std de C++. Cuando se detecta un delimitador (cualquier carácter separador de palabras), se inicia un procedimiento que lee la pila, imprime la palabra separada por sílabas y acumula en una variable unsigned el número de sílabas.

2. **Determinación de acentuación de última palabra:**

Una vez separadas y contabilizadas las sílabas, el análisis métrico indica que se debe contemplar la acentuación de la última palabra, ya que según el tipo puede sumar ±1 sílaba al verso. Por ello, creamos una regla especial cuando se consume un salto de línea, que invoca a un procedimiento que, si la línea contenía al menos una palabra, extrae la última (que es la que se encuentra, separada por sílabas, en la pila), y busca un carácter con tilde. Si lo encuentra, determina la acentuación en función de en cuál sílaba está. Si no, determina la acentuación en función de la última letra de la palabra.

3. **Contemplar sinalefa, diéresis e H/Y:**

En este punto el programa ya presenta resultados convincentes, pero en la mayoría de los versos pasados como prueba el resultado, aunque cercano, no es exacto. Esto es debido a la licencia poética conocida como sinalefa, que une el final de una palabra y el inicio de la siguiente si los caracteres que los une son vocales. Para ello simplemente se crea una expresión regular que es contemplada dentro del núcleo de la sílaba. El otro problema radica en la naturaleza flexible de la H e Y, pues por su fonética son, en muchos casos, contempladas como vocales. Por ello, dentro de las expresiones regulares de sinalefa y diptongo se contempla su posible aparición, pero solo en posiciones específicas para poder diferenciar los momentos cuando son contempladas como consonantes.

Por ejemplo, a/ho/ra (h se funde con la vocal), Al/ham/bra (h actúa como consonante).

Para contemplar el rompimiento de diptongos mediante diéresis solo fue necesario considerarlas vocales abiertas.

No se optó por incluir ninguna otra licencia poética porque las restantes no siguen ningún patrón de uso, por lo que no hay forma de determinar regularmente si se utilizó o no.

4. **Reporte de resultados:**

Finalmente, solo queda la salida adecuada del análisis realizado, consistiendo en la separación en sílabas métricas de cada uno de los versos y el reporte de la cantidad que lo conforman, contemplando el posible ±1 según la acentuación de la última palabra. Además, se decidió proporcionar un mensaje final que indica si la totalidad de los versos provistos son de arte mayor (más de 8 sílabas), arte menor (8 o menos sílabas), o hay presente de ambos tipos. Además, si todos los versos son de igual métrica (mismo número de sílabas), un mensaje adicional indica que todos los versos son de esa cantidad de sílabas, proporcionando una capa adicional de especificidad.

Detalles de implementación

Si bien en el código está comentado la mayoría de detalles, se proporciona un breve resumen del código.

Estructuras de datos hacen falta muy pocas, tan solo un vector para acumular las sílabas de cada palabra, un vector para acumular la cantidad de sílabas de cada verso, un contador de sílabas, un booleano para identificar una nueva palabra y un string std para la impresión de resultados.

Procedimientos tan solo hacen falta tres:

1. palabraPorSilabas(const char * silaba): apila todos los token sílaba extraídos de una palabra.
2. ultimaPalabra(): determina la acentuación de la última palabra del verso
3. finDeVerso(): ignora líneas vacías.

La complejidad radica en las expresiones regulares y reglas, requiriendo 9 reglas distintas para contemplar todas las posibles codas (dado que existen pares de consonantes como ch, pr, cl que no se pueden separar). Además, la herramienta Flex++ no soporta caracteres Unicode, pero es necesario por la naturaleza del problema conservar los caracteres especiales (vocales acentuadas, ñ). Para solucionarlo, hacemos uso de la posibilidad de insertar datos en hexadecimal directamente y especificamos los códigos UTF-8 de estos caracteres, los cuales sí procesa adecuadamente, mientras que el compilador de C++ los reinterpreta como caracteres para la salida.

```
/* Flex++ no reconoce caracteres Unicode, como lo son las vocales acentuadas, pero acepta parámetros binarios con       */
/* tal de que venga definido en 8 bits. Por tanto, se especifican dichos caracteres a partir de su código hexadecimal  */
/* asociado en la tabla Unicode. El prefijo \x especifica parámetros en hexadecimal                                    */
/* En orden de aparición: ÁÉÍÓÚáéíóúÍÜüÜü (en vocales abiertas), Ññ (en consonantes)                                    */
VOCAL_ABIERTA   [AEOaeo]|\xc3[\x81\x89\x8d\x93\x9a\xa1\xa9\xad\xb3\xba\x8f\x9c\xaf\xbc]
```

Uso de código hexadecimal para definir vocales acentuadas.

Sin duda, la complejidad de definir adecuadamente las codas, restringido tanto por las excepciones y normas del idioma como de la poca flexibilidad de uso del metacaracter / en Flex hizo que la mayor carga de complejidad estuviera contenida allí.

```
/* REGLA 1: Coda para consonante cualquiera excepto aquellas que preceden una agregación.   */
/* REGLA 2: Coda para C (excluye H, L, R)                                                   */
/* REGLA 3: Coda para D y R (excluye R)                                                     */
/* REGLA 4: Coda para L (excluye L)                                                         */
/* REGLA 5: Coda para S (excluye H)                                                         */
/* REGLA 6: Coda para B, F, G, K, P, T (excluye L, R)                                       */
/* REGLA 7: Coda vacía                                                                      */
/* REGLA 8: Coda seguida de S (solo dentro de una palabra)                                  */
/* REGLA 9: Coda de múltiples consonantes (solo al final de palabra)                        */
/* REGLA 10: Separación entre palabras de un mismo verso                                    */
/* REGLA 11: Salto de línea. Separación entre versos. Solo actúa ante versos no vacíos      */
/* REGLA 12: Regla por defecto. Sobreescribe ECHO                                           */
%%          /* Reglas */


{SILABA}{C_SIN_AGGS}/({DELIMITADORES}|{CONSONANTE}|\n)      { palabraPorSilabas(YYText()); }

{SILABA}[Cc]/({DELIMITADORES}|{C_SIN_L_R_H}|\n)             { palabraPorSilabas(YYText()); }

{SILABA}[DRdr]/({DELIMITADORES}|{C_SIN_R}|\n)               { palabraPorSilabas(YYText()); }

{SILABA}[Ll]/({DELIMITADORES}|{C_SIN_L}|\n)                 { palabraPorSilabas(YYText()); }

{SILABA}[Ss]/({DELIMITADORES}|{C_SIN_H}|\n)                 { palabraPorSilabas(YYText()); }

{SILABA}{AGG_COMUN}/({DELIMITADORES}|{C_SIN_L_R}|\n)        { palabraPorSilabas(YYText()); }

{SILABA}                                                    { palabraPorSilabas(YYText()); }

{SILABA}{CONSONANTE}[Ss]/{CONSONANTE}                       { palabraPorSilabas(YYText()); }

{SILABA}{CONSONANTE}+/({DELIMITADORES}|\n)                  { palabraPorSilabas(YYText()); }

{DELIMITADORES}+                                            { nueva_palabra = true; }

\n                                                          { finDeVerso(); }

.                                                           {}
```

Definición de las 12 reglas involucradas. Desde la 1 hasta la 9 son la misma función pero contemplando distintas codas.

Se proporciona un Makefile para facilitar el proceso de compilación, así como dos ficheros .txt para prueba (el Makefile toma uno para llevar a cabo una prueba por defecto). Importante mencionar el uso del flag -w en el precompilado de Flex++, para evitar la salida de advertencias en pantalla que genera el uso del metacaracter /, aún cuando está adecuadamente definido para no generar ningún error.

```
javs@Nemo:~/Documents/Manufactura Cristiana/Flex$ make
Em/pie/za a/vi/vir/y em/pie/za/                 Número de sílabas métricas: 8 + 0 = 8
a/mo/rir/de/pun/ta a/pun/ta/                    Número de sílabas métricas: 8 + 0 = 8
le/van/tan/do/la/cor/te/za/                     Número de sílabas métricas: 8 + 0 = 8
de/su/ma/dre/con/la/yun/ta/                     Número de sílabas métricas: 8 + 0 = 8

Ay/cuan/do/los/hi/jos/mue/ren/                  Número de sílabas métricas: 8 + 0 = 8
ro/sas/tem/pra/nas/de a/bril/                   Número de sílabas métricas: 7 + 1 = 8
de/la/ma/dre el/tier/no/llan/to/                Número de sílabas métricas: 8 + 0 = 8
ve/la/su e/ter/no/dor/mir/                      Número de sílabas métricas: 7 + 1 = 8

El poema contiene solamente versos de arte menor.
En específico, todos sus versos son de 8 sílabas
javs@Nemo:~/Documents/Manufactura Cristiana/Flex$
```

Ejecución de Makefile y prueba asociada. El poema leído es octosílabo en su totalidad.

Conclusiones

Si bien el problema fue atacado con cierto escepticismo sobre si era posible solucionarlo en su totalidad, dada la existencia de licencias poéticas y flexibilidad en la poesía difícilmente regularizable, es posible obtener suficiente material para que, con algo de paciencia, se satisfaga exitosamente. Por supuesto, no se descarta en lo absoluto la existencia de versos o palabras que MedidorMetrica no contabilice adecuadamente, pero para una aproximación inicial es una solución sumamente satisfactoria.

Si bien la utilidad de esta aplicación es reducida, es un sólido pilar para la construcción de herramientas de corrección de sintaxis especializadas a la poesía, un universo al que, quizás por las diferencias estilísticas entre poetas e informáticos, no ha sido profundizado.

Bibliografía

Información sobre Flex
http://dinosaur.compilertools.net/flex/manpage.html
Información sobre C++
www.cplusplus.com/
Tabla de códigos UTF-8 en hexadecimal
https://www.utf8-chartable.de/
Información sobre gramática, métrica y ortografía
https://www.sílabas.com/regalas/separar_en_s%C3%ADlabas
https://www.ejemplos.co/100-ejemplos-de-palabras-agudas-graves-y-esdrujulas/
http://www.rinconcastellano.com/tl/metrica.html#
https://www.educa.jcyl.es/educacyl/cm/gallery/recursos_educativa/metrica/pdf/2_versos.pdf
https://es.wikipedia.org/wiki/S%C3%ADlaba#F%C3%B3rmula_sil%C3%A1bica
https://poesiavirtual.com/index.php?ir=reglas/medida.html
https://es.wikipedia.org/wiki/Sinalefa
https://es.wikipedia.org/wiki/Verso_de_arte_mayor
https://es.wikipedia.org/wiki/Verso_de_arte_menor