# Tratamiento previo de textos redactados en castellano

J.A.Mañas

Escuela Técnica Superior de Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

# Tratamiento Previo de Textos Redactados en Castellano

## RESUMEN

El procesamiento de textos redactados en lengua castellana por medio de procesadores desarrollados para lengua inglesa presenta una serie de inconvenientes que limitan su utilización. En el presente trabajo se plantean sus limitaciones y se proponen algunas extensiones en forma de filtro previo para operar con el procesador de textos **nroff** del sistema **UNIX.**[1] Las posibilidades de este filtro incluyen partición automática de palabras a fin de renglón y una cierta comodidad a la hora de introducir vocales acentuadas, con diéresis, la letra "ñ" y símbolos de puntuación como "¿" y "¡". Se describe un eficiente algoritmo de partición junto con algunos criterios para mejorar la apariencia estética del texto. Se caracteriza por ser extremadamente rápido, retrasando escasamente la producción de texto para impresión.

1. UNIX es una marca registrada por los Laboratorios Bell.

| | |
|---|---|
| Desocupado lector, sin juramento me podras creer que quisiera que este libro, como hijo del entendimiento, fuera el mas hermoso, el mas gallardo y mas discreto que pudiera imaginarse. Pero no he podido yo contravenir a la orden de naturaleza; que en ella cada cosa engendra su semejante. Y asi, ?que podra engendrar el esteril y mal cultivado ingenio mio sino la historia de un hijo seco, avellanado, antojadizo, y lleno de pensamientos | varios y nunca imaginados de otro alguno, bien como quien se engendro en una carcel, donde toda incomodidad tiene su asiento y donde todo triste ruido hace su habitacion? El sosiego, el lugar apacible, la amenidad de los campos, la serenidad de los cielos, el murmurar de las fuentes, la quietud del espiritu, son grande parte para que las musas mas esteriles se muestren fecundas y ofrezcan partos al mundo que le colmen de maravilla y de contento. . . . |

## 1. Introducción

Ciertamente malparado resultó D. Miguel de Cervantes tras éste su primer lance con un procesador de textos. Pues pese a sus múltiples e innegables ventajas, su avanzado diseño, y ser muy moderno, también era, ¿cómo no?, norteamericano.

Y es que la amplia utilización de procesadores de textos desarrollados en paises anglosajones causa una serie de problemas a la hora de redactar documentos en lengua castellana. Fundamentalmente son tres los problemas:

a. Incorrecta partición de palabras a fin de renglón ("hyphenation") siguiendo reglas inglesas que no se corresponden con las normas castellanas.

b. Compleja introducción de algunos caracteres como vocales acentuadas, con diéresis, y la letra "ñ", sin correspondencia con el estandar aceptado en máquinas de escribir.

c. Carencia de símbolos de puntuación propios del castellano como "¿" y "¡".

Siendo conveniente por facilidad y compatibilidad la utilización de dichos procesadores de textos de procedencia extranjera, surge la necesidad o conveniencia de intentar adaptarlos a nuestros requisitos con el mínimo esfuerzo posible. Una vía es modificar los programas fuente, vía compleja, arriesgada y no siempre posible. Otra solución, que es la adoptada por el filtro **ftc**, consiste en filtrar previamente el texto, modificando las partes convenientes para adaptarlo a los requisitos del procesador. Esta es además la forma más recomendada para trabajar en UNIX. Aunque presenta algunos inconvenientes, son estos de tan escasa magnitud que se ven ampliamente compensados por las ventajas obtenidas.

## 2. Objetivos

**ftc** persigue los siguientes objetivos:

a. Partición automática de las palabras a fin de renglón.

b. Cómoda introducción de los acentos siguiendo el formato aceptado de las máquinas de escribir: "'o".

c. Cómoda introducción de las letras con diéresis como "ü", con la secuencia ":u".

d. Cómoda introducción de la letra ñ, con la secuencia "'n".

e. Cómoda introducción de los símbolos "¿" y "¡", utilizando las secuencias "'?" y "'!".

f. Conectable a un procesador convencional como el **nroff** [Ossa-76] de **UNIX**, fomando parte de una cadena (pipe) de funciones.

g. Fácil uso, con escasas órdenes nuevas.

h. Debe ser rápido.

## 3. Reglas de partición de palabras

A fin de mejorar la presentación y aprovechar el espacio disponible de papel, la Real Academia Española [Real-73] permite la introducción del signo ortográfico "-" (guión) al fin del reglón que termina con parte de una palabra cuya otra parte, por no caber en él, se ha de escribir en el renglón siguiente. Las palabras se parten a conveniencia del escritor debiendo respetarse las siguientes reglas:

1. Cuando al fin del renglón no cupiere un vocablo entero, se escribirá sólo una parte, la cual siempre ha de formar sílaba cabal. Así, las palabras con-ca-vi-dad, pro-tes-ta, sub-si-guien-te, podrán dividirse a fin de renglón por donde señalan los guiones que van interpuestos en dichas voces, mas no de otra suerte.

2. Esto no obstante, cuando un compuesto sea claramente analizable como formado de palabras que por sí solas tienen uso en la lengua, o de una de estas palabras y un prefijo, será potestativo dividir el compuesto separando sus componentes, aunque no coincida la división con el silabeo del compuesto. Así, podrá dividirse no-sotros o nos-otros, de-samparo o des-amparo.

3. Como cualquier diptongo o triptongo no forma sino una sílaba, no deben dividirse las letras que lo componen. Así, se escribirá gra-cio-so, tiem-po, no-ti-ciáis, a-ve-ri-güéis.

4. Cuando la primera o última sílaba de una palabra fuere una vocal, se evitará poner esta letra sola en fin o en principio de línea.

5. Cuando al dividir una palabra por sus sílabas haya de quedar en principio de línea una "h" precedida de consonante, se dejará esta al fin del renglón y se comenzará el siguiente con la "h": al-haraca, in-humación, clor-hidrato, des-hidratar.

6. En las dicciones compuestas de preposición castellana o latina, cuando después de ella viene una "s" y otra consonante además, como en constante, inspirar, obstar, perspicacia, se han de dividir las sílabas agregando la "s" a la preposición y escribiendo, por consiguiente, cons-tan-te, ins-pi-rar, obs-tar, pers-pi-ca-cia.

7. La "ch" y la "ll", letras simples en su pronunciación y dobles en su figura, no se desunirán jamás. Así, co-che y ca-lle se dividirán como aquí se ve. La erre ("rr") se halla en el mismo caso, y por ello debe evitarse separar los dos signos de que consta, que habrán de ponerse de esta manera: ca-rre-ta, pe-rro.

Nótese la considerable diferencia con la lengua inglesa, en
a cual la partición de palabras entre renglones se realiza de
cuerdo con criterios silábicos (unidades de sonido), estructura-
es (unidades de significado) y de significado (palabras compues-
as) [Horn-74]. Estos son criterios mínimos que se aplican junto
otras varias reglas adicionales. La aplicación de estas reglas
; orientativa; pero no garantiza una correcta[2] partición. En
tima instancia, el escritor debe remitirse a un buen dicciona-
o. Es sintomático que en el sistema UNIX exista un filtro
amado hyphen, encargado de entresacar las palabras truncadas en
texto para que el escritor pueda verificar (proof-reading) la
rrección de las mismas.

---

. El concepto de corrección en inglés es bastante relativo pues, al carecer de una Academia de la Lengua, hay que referirse a alguna autoridad ampliamente aceptada. Típicamente, a la Universidad de Oxford.

El castellano es más sistemático. La regla 1 nos indica un procedimiento de trabajo consistente en localizar las sílabas que componen una palabra a efectos de saber por donde partir. Dícese sílaba al "sónido o sonidos articulados que constituyen un solo núcleo fónico entre dos depresiones sucesivas de la emisión de voz".

El conjunto de sílabas de la lengua castellana es muy elevado; pero afortunadamente se construyen de una forma sistemática. Gracias a ello podremos olvidarnos de utilizar un diccionario de sílabas posibles con las costosas operaciones de búsqueda que esto implica. Desarrollaremos en su lugar un algoritmo capaz de identificar rápidamente los puntos de ruptura legales. Previamente veamos cómo se forman las sílabas.

En primer lugar, una sílaba consta de un núcleo vocal formado por:

- una vocal: abierta: a e o
  cerrada: i u
  (semivocal: y )

- un **diptongo**: ai au ei eu oi ou
  ia ua ie ue io uo
  (ay) (ey) (oy)
  iu ui (uy)

- un triptongo: iai iei iau
  uai uei uau
  (uay) (uey)

Diptongos y triptongos se rompen si el acento tónico recae en alguna de las vocales cerradas (i, u), excepto en los casos "ui" e "iu", que siguen constituyendo diptongo. Si el acento cae en la vocal abierta (a, e, o), el diptongo o triptongo permanece.

Las consonantes se agrupan en torno al núcleo vocálico de forma que éste siempre atrae a la consonante inmediatamente a la izquierda. En ciertos casos, incluso atrae a las dos consonantes inmediatamente a la izquierda. Esto ocurre exclusivamente con los llamados **pares de consonantes** que son los siguientes:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| bl | cl | | fl | gl | kl | pl | tl[3] |
| br | cr | dr | fr | gr | kr | pr | tr |

3. Hay cierta ambigüedad en la corrección del par "tl" que aparece en palabras como atlas o atleta. En lo que sigue hemos preferido desestimarlo como par indivisible debido a lo afectado de su pronunciación. No obstante, esto depende de los hábitos lingüísticos de cada región. Es muy frecuente en México la aparición del grupo "tl" a principio de palabra. Este se tratará como las anomalías en "psi-" que se comentan más adelante. Para mayor información consúltese [Real-73].

Algunos diccionarios de cierto prestigio [Smit-79] sugieren una regla más simple, cual es considerar par de consonantes a cualquier conjunto de dos consonantes tal que la segunda es "l" o "r". Esto no es muy exacto, como puede apreciarse en los siguientes ejemplos: ved-lo, al-rededor, en-lazar, en-roscar, Car-los, is-la, des-ratizar, haz-lo.

Las demás consonantes se agrupan siempre al núcleo vocálico inmediato por la izquierda. En particular, esto ocurre con los grupos de tres o más consonantes. Esta parece una simplificación válida de la sexta regla de partición. La regla permite tratar sistemáticamente las consonantes que terminan palabra.

Una "h" intercalada en un diptongo es muda y no lo destruye. Tal es el caso de de-sahu-ciar. De hecho la "h" puede considerarse a todos los efectos inexistente a efectos de formar sílabas. Tal es el caso de a-dhe-rir, a-lhe-li.[4]

En lo referente a la "h" existe una diferencia entre la regla de silabeado recién vista y la regla de partición [5].

La "y" se considera semivocal si y solo si se encuentra al final de la palabra. En los demás casos es consonante.

La letra "u" aparece en ciertas posiciones especiales:

que qui gue gui

en las que es muda y puede ignorarse a efectos de composición de sílabas, es decir, equivale a

qe qi ge gi

Casos aparte los constituyen

güe güi

en los que la "u" se comporta como una vocal normal, formando un diptongo con la siguiente.

Palabras heredadas de otras lenguas introducen anomalías en las reglas anteriores. Tal es el caso de las griegas

gnomo, mnemónico, psicología

---

4. En ciertas ocasiones muy especiales, la partición tiene en cuenta la "h", no debido a su grafía, si no a su pronunciación. Tal es el caso de des-hierbar y des-huesar. Debido a su extrema rareza, estos casos no se consideraran en el algoritmo de silabeado, siendo quizás su único error. En cualquier caso, el algoritmo de partición es correcto.

La Real Academia [Real-84] ya admite grafías más simples como

nomo, nemónico, sicología

aunque todavía recomienda la primera forma. En ciertos casos, como "gnosis", ni tan siquiera admite la forma simple.

Otras irregularidades proceden del inglés, como "whisky". Aunque aceptada, se recomienda para élla la grafía "güisqui".

## 4. Algoritmo de partición

Las reglas anteriores sugieren un algoritmo directo consistente en identificar los grupos vocálicos y a partir de ellos agrupar las consonantes hasta tener todas la sílabas. Pero mucho más rápido y simple resulta buscar puntos de ruptura, que por ende delimitan sílabas. Este pequeño cambio de punto de interés permite desarrollar un algoritmo, no solo eficaz, sino incluso extremadamente rápido.

Previamente, y previendo que trabajaremos con herramientas (computadores) desarrollados para lengua inglesa, tendremos que atenernos a sus reglas, según las cuales la letra "ch" son dos caracteres (igual acontece con las letras "ll" y "rr"), no existiendo las correspondientes consonantes castellanas como caracteres independientes. Además, los caracteres

á é í ó ú ü ñ Ñ

simplemente, no existen, ni se pueden escribir de forma trivial. (Se exceptúan algunos casos concretos de sistemas que introducen estos caracteres a costa de otros del juego ASCII. No está claro qué es peor).

La primera anomalía se soluciona sin mas que añadir a nuestro conjunto de pares de consonantes las "parejas" ch, ll y rr. Esto es conceptualmente incorrecto, pero es operativo gracias a que no existen los verdaderos pares

| | | |
|---|---|---|
| chl | lll | rrl |
| chr | llr | rrr |

La tabla 1 recoge el conjunto de pares de consonantes que aceptará ftc junto con la frecuencia típica de aparición de cada par en textos castellanos. Nótese que, excepto por la "ch", la letra "h" nunca forma par, con objeto de atenernos a la regla 5 de partición de palabras entre renglones. Algunos pares son ciertamente muy raros. Tal es el caso de "kl" y "kr" que no aparecen sino en palabras de procedencia extranjera como "krausismo".

**TABLA 1.** Pares de consonantes, frecuencias típicas[5].

| | b | c | d | f | g | k | l | p | r | t |
|---|---|---|---|---|---|---|---|---|---|---|
| h | – | 4.7 | – | – | – | – | – | – | – | – |
| l | 9.8 | 2.3 | – | 0. | 0.7 | 0. | 7.1 | 12.2 | – | – |
| r | 7.6 | 5.5 | 3.4 | 0.2 | 8.8 | 0. | – | 8.1 | 6.6 | 22.9 |

(– significa par incorrecto)

Los pares de consonantes gozan de la propiedad de ser siempre pares. Es decir, si bien las reglas de silabeado agrupan los pares cuando aparecen justo antes de vocal o grupo vocálico, lo cierto es que estos pares no aparecen sino en dichas posiciones, por lo cual nunca se ven desplazados ni total ni parcialmente a formar sílaba con el grupo vocálico anterior.

La carencia de ciertos caracteres castellanos en el conjunto ASCII utilizado por la mayoría de los computadores de origen anglosajón, nos obliga a elegir algún par de caracteres suficientemente fáciles de recordar y rápidos de escribir. Podrían ser

'a 'e 'i 'o 'u :u 'n 'N '? '!

A efectos de formación de sílabas,

á equivale a a (vocal abierta)
é equivale a a
í equivale a a
ó equivale a a
ú equivale a a
ü equivale a u (vocal cerrada)
ñ equivale a n (consonante)

La equivalencia significa que se puede trabajar con la forma equivalente (derecha) sin introducir errores en la partición. Nótese como cualquier vocal abierta, acentuada o no, equivale a la "a". Igual ocurre con cualquier vocal cerrada acentuada, que también es equivalente a la abierta "a". Notese igualmente como se elimina la diéresis, símbolo puramente ortográfico. Por último, la letra "ñ" se hace equivalente a la "n" ya que ambas tienen iguales propiedades (en cuanto a ser consonantes y formar los mismos pares (ver tabla 1).

A efectos del algoritmo, los triptongos se consideran pares consecutivos y superpuestos de diptongos. Ya que éstos no pueden

---

5. Las frecuencias que aparecen en esta tabla, así como otras que constan en tablas posteriores son simplemente valores medios típicos que, obviamente, están sujetos a una fuerte dispersión, dependiendo del tema, del autor, del estilo, etc., etc. Los valores indicados deben tomarse a título indicativo.

dividirse, tampoco quedarán divididos aquellos. Es más, las "u" mudas tras "q" y "g" se considerarán como la vocal "u", formando un diptongo (ficticio) o triptongo con la vocal siguiente. Aparecen unos "triptongos" absolutamente incorrectos gramaticalmente como

uie en quien

Los trataremos como un par de diptongos ("ui" solapado con "ie") sin por ello dar lugar a error alguno de partición.

La "y" tiene en general carácter consonántico, excepto a fin de palabra. A efectos del análisis silábico es indiferente que a final de palabra se trate como semivocal (formando un diptongo) o como consonante (que al no ser atraída por otro núcleo vocálico más a la derecha, se asocia al inmediato izquierdo). Un caso complejo lo constituyen las voces, ya en desuso, del tipo doyle, que introducen una "y" con carácter de semivocal en medio de la palabra. El algoritmo la tratará como consonante y, puesto que no existe el par "yl", resultará asociada a la sílaba izquierda: doy-le, consiguiéndose la partición correcta. Así, la "y" se trata siempre como consonante.

La segunda regla para partir palabras nos decía que no debe quedar una sílaba formada por una única letra (forzosamente una vocal) a final de renglón. Podemos releer la regla como que sólo se considera la posibilidad de partir en sílabas a partir de la segunda letra.

En la práctica conviene forzar algo más la regla y solo considerar la posibilidad de partir en sílabas a partir de un cierto umbral. Este jamás será inferior a dos caracteres. Pero un valor más práctico es 4, evitando muchas particiones, que siempre afean el texto, sin desaprovechar mucho espacio (esto es, sin introducir demasiados blancos de relleno). El presente texto se ha tratado con umbral 4. En lo que sigue nos referiremos a este umbral, que corre a cargo del usuario decidir cuánto debe valer (por defecto, 4; y en todo caso 2 o más).

La segunda regla nos dice asimismo que debe evitarse dejar para el siguiente renglón una sílaba formada por una sola letra. Además, no es útil, pues en el renglón primero ocupa tanto una letra como un guión y en el segundo gastaríamos un lugar para la letra y otro para el blanco de separación.

Podría aplicarse de nuevo el criterio de umbral a fin (derecha) de palabra; pero experiementalmente no se aprecia mejora alguna. Antes todo lo contrario. Además, se complica ligeramente el algoritmo. No se aprecia mejora pues al evitar partir una palabra casi al final, dentro de la zona prohibida por el umbral, obligamos a partirla un poco antes. Aparece igual el feo guión y tenemos que introducir más blancos de relleno. Si los umbrales izquierdo y derecho se solapan, las palabras acaban siendo efectivamente indivisibles, lo que no tiene mayor interés que evitar completamente la aparición de guiones. Si este es el objetivo perseguido, basta imponer un alto umbral izquierdo. La única utilidad de imponer un umbral derecho superior a 2 sería evitar

sílabas cortas a la izquierda del siguiente renglón; pero estas sílabas no causan ningún mal efecto estético. Por todo lo anterior, se desestima un umbral derecho superior al mínimo legal de 2. Cuando hablemos de umbral, siempre nos referiremos a umbral izquierdo.

Resumiendo, la segunda regla implica una diferencia entre silabear y partir entre renglones

| silabeado | partición | |
|---|---|---|
| a-ma-rí-a | ama-ría | (umbral < 4) |
| to-do | todo | (umbral > 2) |

Aunque el algoritmo se desarrolla en primer lugar con esta característica, es fácilmente transformable para separar <u>todas</u> las sílabas. (Ver más adelante).

Con las consideraciones anteriores, resulta que basta examinar 3 caracteres consecutivos de la palabra para saber si se puede partir entre el primero y el segundo. Es decir, para conocer si la palabra "catalina" puede partirse entre la "a" y la "t", basta estudiar el grupo "ata". Esto garantiza automáticamente un umbral derecho de 2 caracteres.

Para conocer todos los puntos de ruptura de la palabra "pulpo" a partir de un <u>umbral</u> de 2, consideraremos los grupos

ulp lpo

Un umbral izquierdo superior no implica sino retrasar el punto donde se considera el primer grupo. Por ejemplo, con umbral 3, para averiguar todos los puntos de ruptura de "amortigüéis", estudiaremos

ort rti tig igu gua uai ais

donde no se considera el grupo "mor". Nótese el tratamiento de ":u" como "u" y de "é" como "a".

Un último ejemplo, "vahído", con umbral 2:

aha had ado

En cada grupo de 3 caracteres se actuará de acuerdo con la primera regla aplicable de la tabla 2, recorriéndola de arriba abajo.

TABLA 2. Reglas de partición

| | esquema | acción | ejemplos | fr |
|---|---|---|---|---|
| 1 | a h i | NO | desa-huciar, pro-hibir | 0. |
| 2 | i h a | NO | bu-hardilla | 0. |
| 3 | v c v | SI | to-do, bú-ho | 26.9 |
| 4 | l p p | SI | ca-rro, co-pla, ca-lle, pon-cho | 3.3 |
| 5 | p p l | NO | acar-rear | 3.8 |
| 6 | c c v | SI | téc-nica, cons-ta | 11.9 |
| 7 | a a l | SI | re-al, to-alla | .7 |
| 8 | demás casos | NO | | 53.3 |
| | | | | 100. |

clave de interpretación:

a -> vocal abierta o acentuada
i -> vocal cerrada
v -> vocal
h -> letra h
pp -> par de consonantes (tabla 1)
c -> consonante cualquiera
l -> letra cualquiera

fr -> frecuencia típica de aparición

El procedimiento recuerda al utilizado por el procesador TEX [Knut-79] que introduce varios esquemas (de distintos tamaños), varios pares indivisibles de consonantes y un diccionario de excepciones relativamente largo. Es de destacar la extrema simplicidad de la tabla 2 y la total ausencia de diccionario de excepciones.

Es fácil seguir sobre la tabla 2 las reglas citadas anteriormente. Sólo merece destacarse el tratamiento (1,2) de la "h" muda para detectar diptongos e impedir algunas de las rupturas propuestas por la regla (3) al considerar la "h" como consonante. Actualmente, el autor no conoce palabra alguna a la que aplicar la regla (2) que sólo se ha mantenido en la tabla por un prurito teórico. Asi mismo, la regla (5) impide que la (6) provoque la ruptura de pares de consonantes. En principio, la regla (5) utilizaría el esquema <p p v>, un par de consonantes antes de vocal; pero como ya dijimos anteriormente, los pares de consonantes sólo se dan antes de vocal (salvo error ortográfico) y basta buscar el esquema <p p l>. Aún se puede ir más lejos e imponer que un par de consonantes de la tabla 1 jamás pueda partirse. El programa **ftc** aplica el esquema <p p l> por ser el más simple.

La regla (7), aunque perfectamente correcta, no es siempre apreciada por todos los usuarios, que alegan motivos estéticos. Nuestra experiencia es que puede eliminarse, acelerando el algoritmo ligeramente (nótese la elevada frecuencia relativa de la regla (8)) sin perderse muchas oportunidades de partición (baja

frecuencia relativa de la regla (7)). Muchos usuarios agradecerían sinceramente su eliminación. Incluso alguna gramática [Smit-79] de reconocido prestigio sugiere que es incorrecto aplicar la regla (7).

Se presentan a continuación algunos ejemplos, indicando los grupos considerados, las reglas aplicadas y el resultado obtenido:

```
"catalina" (umbral = 2)  ->  ca-ta-li-na
        ata  tal  ali  lin  ina
         3    8    3    8    3

"catalina" (umbral = 3)  ->  cata-li-na
        tal  ali  lin  ina
         8    3    8    3

"pulpo" (umbral = 2)  ->  pul-po
        ulp  lpo
         8    6

"amortigüéis" (umbral = 2)  ->  amor-ti-güéis
        mor  ort  rti  tig  igu  gua  uai  ais
         8    8    6    8    3    8    8    8

"vahído" (umbral = 2)  ->  va-hí-do
        aha  had  ado
         3    8    3

"transgredir" (umbral = 2)  ->  trans-gre-dir
        ran  ans  nsg  sgr  gre  red  edi  dir
         8    8    8    4    5    8    3    8

"desahuciar" (umbral = 2)  ->  de-sahu-ciar
        esa  sah  ahu  huc  uci  cia  iar
         3    8    1    8    3    8    8

"colchón" (umbral = 2)  ->  col-chón
        olc  lch  cha  han
         8    4    5    8

"toalla" (umbral = 2)  ->  toa-lla
        oal  all  lla
         7    4    5

"caíamos"
        aaa  aam  amo  mos
         7    7    3    8
```

## 5. Influencia del umbral en la presentación del texto

Al tomar diferentes valores para el umbral, ¿qué es lo que se gana y qué es lo que se pierde? Con un umbral mayor que 2 a principio (izquierda) de palabra se gana en presentación, pues en lugar de partir una palabra se fuerza a que aparezca íntegra en el renglón siguiente. Al desaparecer el guión, el texto es más

legible y elegante. En particular, se evita el antiestético fenómeno de series de líneas terminadas en guión. A cambio, se desperdicia espacio, pues al mandar una palabra al siguiente renglón hay que rellenar con blancos el anterior.

Intentaremos cuantificar estas apreciaciones, aunque en última instancia la palabra definitiva la tiene el escritor que optará por una decisión acorde con su propio gusto o sentido de la estética.

**TABLA 3.** Aparición de líneas truncadas

| u | A | B | C |
|---|---|---|---|
| 2 | 32.1 | 46.7 | 1.46 |
| 3 | 23.4 | 61.1 | 1.30 |
| 4 | 17.1 | 64.3 | 1.26 |
| 5 | 12.2 | 82.8 | 1.10 |
| 6 | 7.7 | 94.8 | 1.03 |
| 7 | 5.0 | 98.0 | 1.02 |
| 10 | 0.7 | 100.0 | 1.00 |
| 100 | 0.0 | - | - |

La tabla 3 recoge algunas medidas de la aparición de renglones terminados en guión en función del umbral elegido. Concretamente,

A. Porcentaje de renglones terminados en "-".

B. De ellos, porcentaje que aparece aislado.

C. Longitud media de las series de renglones terminados en "-".

Se aprecia como disminuye el número de líneas truncadas (A), así como la frecuencia de cadenas (aspecto reflejado en el aumento de B). La longitud media de las cadenas o series de líneas truncadas tiende a 1. Si no se parte (umbral muy grande, 100), desaparecen las líneas truncadas.

**TABLA 4.** Inserción de blancos

Porcentaje (acumulado) de líneas
en las que se insertan hasta ..i blancos.

| u | med | 0 | ..1 | ..2 | ..3 | ..4 | ..5 | ..6 | ..7 | ..8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1.3 | 34. | 61. | 82. | 94. | 99. | | | | |
| 3 | 1.7 | 29. | 54. | 72. | 84. | 95. | 99. | | | |
| 4 | 2.0 | 26. | 49. | 66. | 78. | 88. | 95. | 98. | 99. | |
| 5 | 2.2 | 24. | 46. | 62. | 75. | 84. | 91. | 95. | 99. | |
| 6 | 2.6 | 21. | 41. | 57. | 71. | 80. | 87. | 92. | 97. | 99. |
| 7 | 2.8 | 19. | 38. | 55. | 67. | 77. | 83. | 88. | 94. | 98. |
| 10 | 3.2 | 17. | 35. | 51. | 63. | 72. | 79. | 84. | 89. | 94. |
| 100 | 3.3 | 17. | 34. | 50. | 62. | 72. | 78. | 83. | 89. | 93. |

A cambio, la tabla 4 nos indica lo que vamos perdiendo. Por una parte, nos indica el número medio de blancos a añadir por línea. Este es un dato interesante que crece lentamente. Pero quizás es más importante conocer la distribución de estos blancos. Por ello se refleja el porcentaje de casos en los que hay que insertar hasta un número dado de blancos. Por ejemplo, para un umbral de 4 apreciamos que hay que insertar por término medio 2.0 blancos por línea. Esto no es tan grave como saber que sólo el 78% de las veces insertaremos 3 o menos blancos. ¿Cuán grave es insertar más de 3 blancos? Depende de la longitud de la línea o, más correctamente, del número de blancos en la línea, ya que estos serán los que utilicemos para insertar junto a ellos los de relleno. Nótese que el número de blancos en una línea no es independiente del umbral de partición. En efecto, al enviar una palabra a otro renglón eliminamos directamente un blanco de expansión, al tiempo que obligamos a la introducción de blancos de relleno.

**TABLA 5.** Reparto de blancos a insertar

| | longitud de la línea | | | | | | |
|---|---|---|---|---|---|---|---|
| u | 50 | 55 | 60 | 65 | 70 | 75 | 80 |
| 2 | 6.8 | 7.6 | 8.3 | 9.1 | 9.9 | 10.7 | 11.6 |
| 3 | 6.7 | 7.4 | 8.2 | 9.0 | 9.8 | 10.6 | 11.5 |
| 4 | 6.6 | 7.4 | 8.1 | 8.9 | 9.7 | 10.5 | 11.4 |
| 5 | 6.5 | 7.3 | 8.0 | 8.8 | 9.6 | 10.4 | 11.2 |
| 6 | 6.4 | 7.2 | 7.9 | 8.7 | 9.5 | 10.4 | 11.2 |
| 7 | 6.4 | 7.1 | 7.9 | 8.7 | 9.4 | 10.3 | 11.1 |
| 10 | 6.3 | 7.0 | 7.7 | 8.5 | 9.3 | 10.2 | 11.0 |
| 100 | 6.2 | 7.0 | 7.7 | 8.5 | 9.3 | 10.1 | 11.0 |

La tabla 5 recoge la variación del número de blancos en una línea (antes de insertar los de relleno) en función del umbral y de la longitud de la línea. Para evitar los "ríos" de blancos que se producen al ensanchar líneas consecutivas en columnas cercanas, y teniendo en cuenta que el algoritmo utilizado por nroff [Ossa-76] consiste en ir insertando blancos alternativamente desde la izquierda y desde la derecha, conviene evitar que muchas líneas incrementen en más de un 30% su número de blancos.

Por ejemplo, para líneas de 65 caracteres, en las que el número de blancos ronda los 9, conviene evitar que en muchos casos se introduzcan más de 3 blancos adicionales.

De todo lo anterior, se ha considerado un buen compromiso el umbral de 4, aunque insistimos en que la última palabra depende de los criterios estéticos del escritor. Las tablas anteriores pueden ayudarle a tomar una decisión más personal.

## 6. Algoritmo de silabeado

Como ya se indicó anteriormente, el algoritmo puede modificarse para partir en sílabas. Este aspecto no es de interés para fto, y sólamente se considera aquí por completar la información.

El silabeado y la partición difieren en que el primero elimina los umbrales (o, si se quiere, los reduce a 1). Por ello se considerarán todos los grupos dentro de una palabra. La tabla 6 recoge los criterios a aplicar, muy similares a los de la tabla 2 con algunas modificaciones.

**TABLA 6.** Reglas de silabeado

| | esquema | acción | ejemplos |
|---|---|---|---|
| 1 | - a h i | NO | a-hijar |
| 2 | - i h a | NO | bu-honero |
| 3 | - v c v | SI | a-mar |
| 4 | - l p p | SI | a-probar |
| 5 | - p p - | NO | p-rohibir |
| 6 | l c c v | SI | |
| 7 | - a a - | SI | mare-o |
| 8 | demás casos | NO | |

En primer lugar, a veces se consideran grupos de 3 (reglas 1, 2, 3 y 4), a veces de 2 (reglas (5 y 7), a veces de 4 (regla 6). Por normalizar, todos los esquemas se plasman sobre 4, indicando con "-" las letras irrelevantes. La regla (6) permite controlar las voces como psicología en las que la "p" es muda y no forma par "ps"; pero forma una sóla sílaba, "psi", de origen griego. Si la regla (6) se rebaja al esquema <- c c v> resulta un silabeado como p-si-co-lo-gí-a.

Para un correcto silabeado es necesario introducir en la tabla de pares de consonantes todos los pares terminados en "h". Así se consigue tratar palabras como a-lhe-lí, a-dhe-rir, etc.

Por último, la "y" a fin de palabra toma caracter semivocálico; pero siempre forma diptongo pues jamás va acentuada (en ese caso se escribiría "i"). El algoritmo siempre impide la separación, aunque lo hace considerándola una consonante a agrupar por la izquierda (regla 8).

## 7. Otras características

**ftc** incorpora la tabla 2 para partición automática. Un texto de entrada se filtra indicándose para cada palabra sus posibles puntos de ruptura. Se utiliza el formato estandar de **nroff** [Ossa-79]. Es decir,

```
catalina  (umbral = 2)  ->  ca\%ta\%li\%na
```

Las vocales acentuadas, etc., reciben el tratamiento correspondiente para sobreimprimir caracteres

```
'a  ->  \o'\'a'
:u  ->  \o'"u'
'n  ->  \o'~n'
'   ->  '
```

Los símbolos de puntuación se traducen en una secuencia aparentemente inocua cual es

```
'?  ->  \o':?'
```

dos puntos sobreimpresos con el símbolo inglés más parecido. Para la correcta impresión de "¿" se requiere una impresora capaz de hacerlo, normalmente con algún caracter o secuencia especial que puede ser fácilmente introducida por un filtro posterior (por ejemplo, con **sed**) que detecte la secuencia dos puntos-retorno-?.

El programa puede modificarse rápidamente para obtener otras traducciones. Esto puede ser de interés si se cambia el caracter indicador de punto de partición (orden .hc en nroff). O si se desean mejores secuencias para escribir, por ejemplo, la "ñ".

Ni que decir que debe eliminarse en mecanismo automático de partición en inglés (orden .nh en nroff).

**ftc** utiliza una sola orden

```
.-c n
```

Aunque su formato recuerda totalmente al de las órdenes nroff, sólo es válida totalmente a la izquierda de la línea. **ftc** procesa el texto estrictamente antes que nroff (ver sección siguiente) y por tanto no son de utilidad alguna las facilidades de éste para reordenar texto (por ejemplo con .de). La línea que así comienza es totalmente eliminada del texto y no le llega a nroff.

El argumento único n siempre debe estar presente y es un número decimal indicando el umbral de aplicación de la tabla 2. Inicialmente vale 4. Normalmente sólo se modifica para evitar el algoritmo. Esto es útil si en el mismo fichero conviven textos en castellano y en inglés, dentro de tablas, ecuaciones, bloques de texto (displays), etc. El algoritmo de partición queda efectivamente inhibido imponiendo un alto umbral (digamos 100).

## 8. Utilización

**ftc** lee de su entrada estandar y escribe en su salida estandar. Tipicamente las secuencias de órdenes para UNIX seran:

1 fichero:

```
ftc mifichero | nroff ...
```

varios ficheros

```
ftc f1 f2 ... fn | nroff ...
```

con tablas

```
ftc mifichero | tbl | nroff ...
```

con ecuaciones

```
ftc mifichero | neqn | nroff ...
```

con tablas y ecuaciones

```
ftc mifichero | tbl | neqn | nroff ...
```

Secuencias análogas con troff.

La salida de estas cadenas suele pasarse por algún filtro que gestione los cambios pertinentes de juegos de caracteres, ateniéndose a la impresora correspondiente.

## 9. Evaluación

**ftc** es muy eficiente en el cumplimiento de su labor de procesado. A efectos de evaluar cuán costoso es utilizarlo, hemos realizado algunas simples medidas sobre dos ficheros de texto. El primero, test, es un fichero con sólo cuatro órdenes al principio:

```
.ll 65
.na
.fi
.nh
```

El resto del fichero carace de orden alguna para **ftc** o **nroff**. Por tanto, el umbral es constantemente 4 y nroff produce una salida continua sin párrafos, ni páginas, ni otros detalles de presentación. test es el peor caso imaginable, pues nroff realiza la minima labor nacesaria y ftc la máxima posible. Para evaluar la magnitud podemos decir que

```
ftc test | nroff | wc
```

arroja una cuenta de 1320 lineas, 13027 palabras y 85512 caracteres.

El segundo programa es el presente articulo, yo. Este es más realista por incluir párrafos, paginado, tablas, etc. Se puede procesar como

```
ftc yo | tbl | nroff | col -fx | facit > listado
```

donde facit es un postprocesador para una impresora FACIT 4510. Procesado por **wc**, aporta 1832 lineas, 6940 palabras y 62798 caracteres.

La tabla 7 resume los valores obtenidos al procesar estos dos ficheros. El tiempo que se indica es la suma del tiempo en modo usuario más el tiempo en modo sistema, según datos aportados por la función time del cshell.

**TABLA 7.** Evaluación de velocidad

| orden | tiempo |
|---|---|
| nroff test > /dev/null | 191.5 s |
| ftc test \| nroff > /dev/null | 211.3 s |
| ftc test > /dev/null | 41.4 s |
| tbl yo \| nroff > /dev/null | 516.3 s |
| ftc yo \| tbl \| nroff > /dev/null | 520.2 s |
| ftc yo > /dev/null | 17.7 s |

Estas medidas justifican la calificación de rápido para **ftc**. yo puede considerarse un caso tipico. Filtrarlo con **ftc** se traduce en un incremento de sólo un 0.8% en el tiempo de proceso. En el caso peor considerado, test, puede suponer hasta un 10% de costo adicional.

A fin de valorar la carga intrínseca de conectar dos programas por un "pipe", se ha medido igualmente el tiempo propio de proceso de **ftc**, con los resultados que se indican en la tabla 7. Es de destacar la carga de trabajo que supone para **ftc** tener que averiguar todos los puntos de ruptura de todas las palabras del texto, como es el caso en test.

## 10. Referencias


[Horn-74] Hornby, A.S., Oxford Advanced Learner's Dictionary of Cuurent English. Oxford University Press, 3rd ed, 1974.

[Knut-79] Knuth, D.E., TEX and Metafont. New Directions in Typesetting. Digital Press, 1979.

[Ossa-76] Ossanna, J.F. nroff/troff user's Manual. Computing Science Technical Report no. 54. Bell Labs 1976.

[Real-73] Real Academia Española. Esbozo de una Nueva Gramática de la Lengua Española. Espasa-Calpe, S.A., Madrid, 1973.

[Real-84] Real Academia Española. Diccionario de la Lengua Española. Vigésima edición, Madrid 1984.

[Smit-79] Smith, C. et al. Diccionario Collins Español-Inglés, Inglés-Español. Grijalbo, Barcelona, 1979.

A. Listado de itc

```
# include <stdio.h>
# include <ctype.h>

FILE *fp;

# define NO      0
# define SI     -1

int     t[14], a[3], b[3],
        it= 0, ia= 0, i, itr,
        nl= 0, umbral= 4,
        i, c,
        izq,
        clave[26]=
          {0,2,2,2,0,2,2,2,1,2,2,2,2,2,0,2,2,2,2,2,1,2,2,2,2,2},
        C    [3] = {NO, NO, SI},
        V    [3] = {SI, SI, NO},
        VA   [3] = {SI, NO, NO},
        VC   [3] = {NO, SI, NO};

int     pp [2][17] =
          {{'t','p','b','g','p','b','l','r','c','c','d','c','g',
            'f','f','k','k'},
           {'r','l','l','r','r','r','l','r','r','h','r','l','l',
            'r','l','l','r'}};

par (c1, c2) int c1, c2; {
        for (i= 0; i < 17; i++)
          if ( (pp[0][i] == c1) && (pp[1][i] == c2) ) return (SI);
        return (NO);
        }

posible () {
        for (i= 0; i < 3; i++) b[i]= clave [a[i] - 'a'];
        if ( C[b[1]] ) {
          if (   (a[1] == 'h') &&  (    (( VA[b[0]] ) && ( VC[b[2]] ))
                                     || (( VC[b[0]] ) && ( VA[b[2]] ))
                                                          return (NO);
              )
          if ( ( V[b[0]] ) && ( V[b[2]] ) )               return (SI);
          if ( ( C[b[2]] ) && par (a[1], a[2]) )          return (SI);
          if ( ( C[b[0]] ) && par (a[0], a[1]) )          return (NO);
          if ( ( C[b[0]] ) && ( V[b[2]] ) )               return (SI);
          } ;
        if ( ( VA[b[0]] ) && ( VA[b[1]] ) )               return (SI);
                                                          return (NO);

        }
```

```
simp (c1, c2) char c1, c2; {
        it--;
        t[it++]= '\';
        t[it++]= 'o';
        t[it++]= ''';
        if (c1 == ''') t[it++]= '\';
        t[it++]= c1;
        t[it++]= c2;
        t[it++]= ''';
        }

filtro () {
        izq= 2;
        while ((t[it++]= c= getc(fp) ) != EOF) {
          if (izq > 0) izq--;
          if (isalpha (c) )
            if (++nl >= umbral) a[ia++]= tolower(c);
            else ;
          else if (c == ''') {
            c= getc(fp);
            switch (tolower(c)) {
              case 'a':
              case 'e':
              case 'i':
              case 'o':
              case 'u':
                if (++nl >= umbral) a[ia++]= 'a';
                simp (''', c);
                break;
              case 'n':
                if (++nl >= umbral) a[ia++]= 'n';
                simp ('~', c);
                break;
              case '?':
              case '!':
                ia= nl= 0;
                simp (':', c);
                break;
              case ''':
                ia= nl= 0;
                break;
              default:
                ungetc (c, fp);
                ia= nl= 0;
              }
            }
```

```
else if (c == ';') {
  c= getc(fp);
  if (tolower(c) == 'u') {
    if (++nl >= umbral) a[ia++]= 'u';
    simp ('"', c);
    }
  else {
    ungetc (c, fp);
    ia= nl= 0;
    };
  }
else {
  ia= nl= 0;
  if (izq && (c == '.') ) {
    t[it++]= getc(fp);
    t[it++]= getc(fp);
    if (  (t[it-2]== '-') && (t[it-1]== 'c') ) {
      fscanf (fp, "%d", &umbral);
      while ( (c= getc(fp)) != '0) /* saltatelo */ ;
      it-= 3;
      }
    }
  if (c == '0) izq= 2;
  }
```

```
        if (ia <= 1) {
          for (i= 0; i < it; i++) putchar (t[i]);
          it= 0;
          }
        else if (ia == 2) itr= it;
        else {                                          /* ia == 3 */
          if (posible()) { putchar ('\'); putchar ('%'); };
          for (i= 0; i < itr; i++) putchar (t[i]);
          for (i= 0; i < it-itr; i++) t[i]= t[itr+i];
          a[0]= a[1];
          a[1]= a[2];
          ia= 2;
          it-= itr;
          itr= it;
          }
        }
      it--;   /* eliminacion de EOF */
      }

main (argc, argv)
int argc;
char *argv[];
      {
      FILE *fopen();

      if (argc == 1) {   /* standard input */
        fp = stdin;
        filtro ();
        }
      else while (--argc > 0)
        if ((fp= fopen (*++argv, "r")) == NULL) {
          fprintf (stderr, "ftc: no puedo abrir %s0, *argv);
          break;
          }
        else {
          filtro ();
          fclose (fp);
          }
      }
```