Ejercicio 3, Apartado 2

Número de grupo: 03

Correos: 1.Eder Tarifa: eder.tarifa@alumnos.upm.es 2.Mario García: mario.gberenguer@alumnos.upm.es 3.Álvaro Hernández: alvaro.hernandezr@alumnos.upm.es 4.David Hernado: david.hernando@alumnos.upm.es

CÓDIGO

Librerias necesarias para la práctica.

```
library(sylly.es)
```

```
## Loading required package: sylly
```

```
## Warning: package 'sylly' was built under R version 4.2.2
```

```
library(stringr)
```

Código para obtener un diccionario reducido. Como no existe palabras que empiecen por mayúscula en el diccionario que debemos usar, solo usamos el filtro de más de 5 letras.

```
dic <- read.table(file='dic_es.txt', encoding='UTF-8')
eliminar <- vector()
for (i in 1:(dim(dic)[1])){
  if (str_count(dic[i,1]) < 5){
    eliminar <- c(eliminar, i)
  }
}
dic <- dic[-eliminar,]
dic_clean <- file(description = "dic_es_clean.txt",open = "wt",
                  blocking = TRUE,encoding = "UTF-8")
writeLines(dic, dic_clean,sep = "\n")
close(dic_clean)
```

Funciones auxiliares para sacar la cadena, la primera sílaba y la última sílaba.

```
#sacamos la última sílaba de la palabra

sacar_ultima_silaba <- function(palabra){
  res <- hyphen(palabra, hyph.pattern="es", min.length = 2, quiet = TRUE)
  silabas <- unlist(strsplit(res@hyphen[["word"]],""))
  numero_silabas = res[[1]]
  guiones_encontrados = 0
  n = 1
  while (guiones_encontrados < numero_silabas - 1){
```

```
    if (silabas[n] == '-'){
      guiones_encontrados = guiones_encontrados + 1
    }
    silabas <- silabas[(n+1):length(silabas)]
  }
  ultima_silaba <- chartr("áéíóú", "aeiou",paste(silabas, collapse = ""))
  return (ultima_silaba)
}

#sacamos la primera sílaba de la palabra
sacar_primera_silaba <- function(palabra){
  res <- hyphen(palabra, hyph.pattern="es", min.length = 2, quiet = TRUE)
  silabas <- unlist(strsplit(res@hyphen[["word"]],""))
  if (res[[1]] == 1){
    primera_silaba = res[[2]]
  }
  else{
    guiones_encontrados <- 0
    n <- 1
    primera_silaba = vector()
    while (guiones_encontrados <  1){
      if (silabas[n] == '-'){
        guiones_encontrados = guiones_encontrados + 1
      }
      else{
        primera_silaba[n] <- silabas[n]
      }
      n = n + 1
    }
    primera_silaba <- chartr("áéíóú", "aeiou", paste(primera_silaba,
                                                     collapse = ""))
  }
  return (primera_silaba)
}

sacar_encadenacion_ganadora <- function(palabra){
  encadenacion <- c(palabra)#vector con la palabra
  for(i in 1:1){#solo queremos una palabra en la encadenación
    ultima_silaba <- sacar_ultima_silaba(palabra)
      if (substr(ultima_silaba, start = 1, stop = 1) == "a"){#vamos a filtrar
        #para no tener que recorrer el diccionario entero
    dic <- lista_a
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "b"){
    dic <- lista_b
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "c"){
    dic <- lista_c
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "d"){
    dic <- lista_d
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "e"){
```

```
 dic <- lista_e
}
if (substr(ultima_silaba, start = 1, stop = 1) == "f"){
  dic <- lista_f
}
if (substr(ultima_silaba, start = 1, stop = 1) == "g"){
  dic <- lista_g
}
if (substr(ultima_silaba, start = 1, stop = 1) == "h"){
  dic <- lista_h
}
if (substr(ultima_silaba, start = 1, stop = 1) == "i"){
  dic <- lista_i
}
if (substr(ultima_silaba, start = 1, stop = 1) == "j"){
  dic <- lista_j
}
if (substr(ultima_silaba, start = 1, stop = 1) == "k"){
  dic <- lista_k
}
if (substr(ultima_silaba, start = 1, stop = 1) == "l"){
  dic <- lista_l
}
if (substr(ultima_silaba, start = 1, stop = 1) == "m"){
  dic <- lista_m
}
if (substr(ultima_silaba, start = 1, stop = 1) == "n"){
 dic <- lista_n
}
if (substr(ultima_silaba, start = 1, stop = 1) == "o"){
  dic <- lista_o
}
if (substr(ultima_silaba, start = 1, stop = 1) == "p"){
  dic <- lista_p
}
if (substr(ultima_silaba, start = 1, stop = 1) == "q"){
  dic <- lista_q
}
if (substr(ultima_silaba, start = 1, stop = 1) == "r"){
  dic <- lista_r
}
if (substr(ultima_silaba, start = 1, stop = 1) == "s"){
    dic <- lista_s
}
if (substr(ultima_silaba, start = 1, stop = 1) == "t"){
    dic <- lista_t
}
if (substr(ultima_silaba, start = 1, stop = 1) == "u"){
  dic <- lista_u
}
if (substr(ultima_silaba, start = 1, stop = 1) == "v"){
  dic <- lista_v
}
```

```
  if (substr(ultima_silaba, start = 1, stop = 1) == "w"){
    dic <- lista_w
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "x"){
    dic <- lista_x
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "y"){
    dic <- lista_y
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "z"){
    dic <- lista_z
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "ñ"){
    dic <- lista_ñ
  }
    dimension <- length(dic)
    encontrado <- FALSE
    #print(ultima_silaba)
    j <- 1
    while((!encontrado) && (j <= dimension)){#solo recorremos el bucle si no
      #encontramos nada
      palabra2 = dic[j]
      palabra_letras <- unlist(strsplit(palabra2, ""))
      primeras_letras <- paste(palabra_letras[1:str_count(ultima_silaba)],
                               collapse = "")
      if ( primeras_letras == ultima_silaba){ #sistema de optimización
        primera_silaba <- sacar_primera_silaba(palabra2)
        if (primera_silaba == ultima_silaba && palabra != palabra2){
          encontrado = TRUE
          encadenacion <- c(encadenacion, palabra2)
        }
      }
      #print(j)
      j = j + 1
    }
    if (j > dimension){
      break
    }

  }
  return (encadenacion)
}
```

Diccionario dado para buscar aquí las palabras Listas con todas las palabras ordenadas según la letra con la que empiezan. Nos permite optimizar mucho el algoritmo

```
dic_or <- read.table(file='dic_es_clean.txt', encoding='UTF-8')

dic=dic_or[,1]

lista_a <-c()
lista_b <-c()
lista_c <-c()
```

```
lista_d <-c()
lista_e <-c()
lista_f <-c()
lista_g <-c()
lista_h <-c()
lista_i <-c()
lista_j <-c()
lista_k <-c()
lista_l <-c()
lista_m <-c()
lista_n <-c()
lista_o <-c()
lista_p <-c()
lista_q <-c()
lista_r <-c()
lista_j <-c()
lista_s <-c()
lista_t <-c()
lista_u <-c()
lista_v <-c()
lista_w <-c()
lista_x <-c()
lista_y <-c()
lista_z <-c()
lista_ñ <- c()

for (i in 1:length(dic)){
  if (substr(dic[i], start = 1, stop = 1) == "a"){
    lista_a <- c(lista_a, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "b"){
    lista_b <- c(lista_b, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "c"){
    lista_c <- c(lista_c, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "d"){
    lista_d <- c(lista_d, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "e"){
    lista_e <- c(lista_e, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "f"){
    lista_f <- c(lista_f, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "g"){
    lista_g <- c(lista_g, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "h"){
    lista_h <- c(lista_h, dic[i])
  }
  if (substr(dic[i], start = 1, stop = 1) == "i"){
    lista_i <- c(lista_i, dic[i])
```

```
}
if (substr(dic[i], start = 1, stop = 1) == "j"){
  lista_j <- c(lista_j, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "k"){
  lista_k <- c(lista_k, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "l"){
  lista_l <- c(lista_l, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "m"){
  lista_m <- c(lista_m, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "n"){
  lista_n <- c(lista_n, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "o"){
  lista_o <- c(lista_o, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "p"){
  lista_p <- c(lista_p, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "q"){
  lista_q <- c(lista_q, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "r"){
  lista_r <- c(lista_r, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "s"){
  lista_s <- c(lista_s, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "t"){
  lista_t <- c(lista_t, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "u"){
  lista_u <- c(lista_u, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "v"){
  lista_v <- c(lista_v, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "w"){
  lista_w <- c(lista_w, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "x"){
  lista_x <- c(lista_x, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "y"){
  lista_y <- c(lista_y, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "z"){
  lista_z <- c(lista_z, dic[i])
}
if (substr(dic[i], start = 1, stop = 1) == "ñ"){
```

```
    lista_ñ <- c(lista_ñ, dic[i])
  }
}
```

Función que inicia el juego. Esta lee las palabras del usuario por consola.

```
#funcion que inicia el juego
juego <- function() {
  jugando=TRUE
  palabra_valida=FALSE
  while (palabra_valida == FALSE){#hasta que introduzcamos una palabra válida
    palabra_input <- readline(prompt = "Introduce una palabra válida -> ")
    if(palabra_input %in% dic){
      palabra_valida = TRUE
    }
    else{
      print("Palabra no válida, introduce otra")
    }
  }

  while (jugando == TRUE){
    encadenacion_global = c(palabra_input)
    palabra_valida=FALSE
    palabra = palabra_input
    ultima_silaba = sacar_ultima_silaba(palabra)
    if (substr(ultima_silaba, start = 1, stop=2) == "rr"){#no existen palabras
      #que comiencen por "rr"
      print(paste("El usuario ha ganado, no existe ninguna palabra que comience por ",ultima_silaba))
      print(paste(encadenacion_global))
      palabra_valida=TRUE
      jugando=FALSE
    }
    else{
    encadenacion <- sacar_encadenacion_eligiendo_ganadoras2(palabra)
    if (encadenacion == "No existe" && jugando){
      print(paste("El usuario ha ganado, no existe ninguna palabra que continue a ", palabra))
      jugando=FALSE
      palabra_valida=TRUE
    }
    else if (length(encadenacion) == 1 && jugando){
        print(paste("La máquina ha ganado, ya que no hay ninguna palabra que continue a", encadenacion))
        print(paste(encadenacion_global, encadenacion))
        jugando = FALSE
        palabra_valida=TRUE
      }
    else { #el ordenador elige una palabra
        palabra_actual <- encadenacion[2]
        silaba=sacar_ultima_silaba(palabra_actual)
        encadenacion_global <- c(encadenacion_global, encadenacion[2])
          print(paste("He elegido la palabra : ", palabra_actual))
        print(paste("Dime una palabra que empieze por la sílaba '",
                    silaba, "'"))
        palabra_valida=FALSE
```

```
      }
    }

    while (palabra_valida == FALSE && jugando == TRUE){#si el juego sigue
      #debemos introducir otra palabra
      palabra_input <- readline(prompt = "Introduce una palabra válida -> ")
      prim_sil <- sacar_primera_silaba(palabra_input)
      if (palabra_input == "Me rindo"){ #puedes rendirte si no se te ocurre
        #alguna palabra
        print("El ordenador ha ganado porque te has rendido")
        print(paste(encadenacion_global))
        jugando = FALSE
      }
      else if((prim_sil == silaba) && (palabra_input %in% dic)){
        palabra_valida = TRUE
      }else{
        print("Palabra no válida, introduce otra")
      }
    }
  }
}
```

Aquí definimos la función para sacar las palabras ganadoras entre las totales que matchean, pero tarda mucho en ejecutar. Codigo ejecutado con mandoble a las 23:28:00 23:39:33 saca como palabra ganadora blefaroplastias.

```
sacar_encadenacion_eligiendo_ganadoras <- function(palabra){
  palabras_match <- c()
  for(i in 1:1){
    dimension = dim(dic_or)[1]
    ultima_silaba <- sacar_ultima_silaba(palabra)
    encontrado <- FALSE
    print(ultima_silaba)
    j <- 1
    while (j <= dimension){
      palabra2 = dic[j]
      palabra_letras <- unlist(strsplit(palabra2, ""))
      primeras_letras <- paste(palabra_letras[1:str_count(ultima_silaba)],
                               collapse = "")
      if ( primeras_letras == ultima_silaba){
        primera_silaba <- sacar_primera_silaba(palabra2)
        if (primera_silaba == ultima_silaba && palabra != palabra2){
          palabras_match <- c(palabras_match, palabra2) #añadimos las palabras
          #a una lista de palabras que matchean
        }
      }
      print(j)
      j = j + 1
    }
    if (j > dimension){
      print("No se puede continuar la encadenación.")
      palabra_elegida = palabra
    }
```

```
    k=1
    encontrado2=FALSE
    palabra_elegida=palabras_match[1]
    while (k <=length(palabras_match) && encontrado2 == FALSE){#buscamos
      #una palabra de la lista que no tenga continuación
      if (length(sacar_encadenacion_ganadora(palabras_match[k])) == 1){
        palabra_elegida = palabras_match[k]
        encontrado2=TRUE
      }
      k = k +1
    }
  }
  return (palabra_elegida)
}
```

Aquí definimos la función que busca las palabras que matcheen con la que me da el usuario. Esta se diferencia de la otra en que busca si la plabra es ganadora nada más obtenerla.

```
sacar_encadenacion_eligiendo_ganadoras2 <- function(palabra){
  palabras_match <- c()
  palabra_elegida=""
  for(i in 1:1){
    ultima_silaba <- sacar_ultima_silaba(palabra)
    encontrado <- FALSE
    print(ultima_silaba)
      if (substr(ultima_silaba, start = 1, stop = 1) == "a"){#filtramos para
        #no tener que recorrer todo el diccionario entero
    dic <- lista_a
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "b"){
    dic <- lista_b
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "c"){
    dic <- lista_c
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "d"){
    dic <- lista_d
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "e"){
   dic <- lista_e
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "f"){
    dic <- lista_f
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "g"){
    dic <- lista_g
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "h"){
    dic <- lista_h
  }
  if (substr(ultima_silaba, start = 1, stop = 1) == "i"){
    dic <- lista_i
  }
```

```
if (substr(ultima_silaba, start = 1, stop = 1) == "j"){
  dic <- lista_j
}
if (substr(ultima_silaba, start = 1, stop = 1) == "k"){
  dic <- lista_k
}
if (substr(ultima_silaba, start = 1, stop = 1) == "l"){
  dic <- lista_l
}
if (substr(ultima_silaba, start = 1, stop = 1) == "m"){
  dic <- lista_m
}
if (substr(ultima_silaba, start = 1, stop = 1) == "n"){
 dic <- lista_n
}
if (substr(ultima_silaba, start = 1, stop = 1) == "o"){
  dic <- lista_o
}
if (substr(ultima_silaba, start = 1, stop = 1) == "p"){
  dic <- lista_p
}
if (substr(ultima_silaba, start = 1, stop = 1) == "q"){
  dic <- lista_q
}
if (substr(ultima_silaba, start = 1, stop = 1) == "r"){
  dic <- lista_r
}
if (substr(ultima_silaba, start = 1, stop = 1) == "s"){
    dic <- lista_s
}
if (substr(ultima_silaba, start = 1, stop = 1) == "t"){
    dic <- lista_t
}
if (substr(ultima_silaba, start = 1, stop = 1) == "u"){
  dic <- lista_u
}
if (substr(ultima_silaba, start = 1, stop = 1) == "v"){
  dic <- lista_v
}
if (substr(ultima_silaba, start = 1, stop = 1) == "w"){
  dic <- lista_w
}
if (substr(ultima_silaba, start = 1, stop = 1) == "x"){
  dic <- lista_x
}
if (substr(ultima_silaba, start = 1, stop = 1) == "y"){
  dic <- lista_y
}
if (substr(ultima_silaba, start = 1, stop = 1) == "z"){
  dic <- lista_z
}
if (substr(ultima_silaba, start = 1, stop = 1) == "ñ"){
  dic <- lista_ñ
```

```
  }
    dimension <- length(dic)
    palabra_ganadora <- FALSE
    j <- 1
    while (j <= dimension && encontrado == FALSE){
      palabra2 = dic[j]
      palabra_letras <- unlist(strsplit(palabra2, ""))
      primeras_letras <- paste(palabra_letras[1:str_count(ultima_silaba)],
                               collapse = "")
      if ( primeras_letras == ultima_silaba){
        primera_silaba <- sacar_primera_silaba(palabra2)
        if (primera_silaba == ultima_silaba && palabra != palabra2){
          palabras_match <- c(palabras_match, palabra2)
          palabra_elegida <- palabras_match[sample(1:length(palabras_match))]
          if (length(sacar_encadenacion_ganadora(palabra2)) == 1){
            encontrado=TRUE
            palabra_ganadora=TRUE
            palabra_elegida=palabra2
          }
        }
      }
      #print(j)
      j = j + 1
      encadenacion <- c(palabra,palabra_elegida)
    }
    if (palabra_ganadora == TRUE){
      encadenacion <- palabra_elegida
    }
    if (j > dimension){
      print("No se puede continuar la encadenación.")
      encadenacion <- "No existe"
    }

  return (encadenacion)
  }
}
```

COMENTARIOS

Contenido: -Función sacar_primera_silaba(palabra): saca la primera sílaba de la palabra -Función sacar_última_silaba(palabra): saca la última sílaba de la palabra -Función sacar_encadenacion_ganadora(palabra): devuelve un vector con la palabra dada y la palabra que hemos encadenado en el caso de que exista -Función sacar_encadenacion_eligiendo_ganadora(palabra): devuelve una encadenación con la palabra que introducimos y la que encuentra la máquina. Si noencuentra solo devuelve la palabra que da la máquina. -Función juego(): comienza el juego. Es donde se piden las palabras. Usamos la función sacar_encadenación_eligiendo_ganadora2 para reducir el tiempo que tarda en sacar cada palabra -Función sacar_encadenación_eligiendo_ganadora(): primer prototipo para sacar encadenación. Tarda mucho en ejecutar por no estar optimizada. -Dataframe dic: contiene todas las palabras que vamos a poder usar. Luego se convierte en una lista. -Lista lista_palabras: contiene una lista de palabras que no se pueden continuar -Lista silabas_perdedoras: contiene una lista con las sílabas que tienen asignadas palabras que no se pueden continuar

Para el ejercicio 2 se nos hacer el juego de las palabras encadenadas con algunas reglas: -Las palabras debían de tener más de 5 letras -Se usarán las sílabas que separe la función hyphen

Para ello se han podido reciclar algunas funciones auxiliares del ejercicio 1, como la función sacar_primera_silaba(palabra) la función sacar_segunda_silaba(palabra). También se ha modificado la función sacar_encadenacion_ganadora(palabra) para que funcione en este caso, ya que el programa no separa bien las sílabas. Por ejemplo, si nosotros no cambiaramos la función y introdujeramos la palabra "arroz", el programa la cuenta como una única sílaba y la función devuelve la propia palabra "arroz", algo que no sería válido en el juego. De la misma forma no vamos a usar nuestro diccionario inicial, sino que vamos a eliminar de él las palabras con menos de 5 letras.

En primer lugar pensamos en hacer una función que comparara todas las palabras del diccionario que teníamos entre sí, para así identificar aquellas que no tenían continuación y meterlas en una lista para evitar calcularlas de nuevo cada vez que buscaramos una palabra. No obstante, esto suponía un gran problema, ya que el tiempo que tardaba en comparar una palabra con todas las del diccionario era increiblemente alto. Conseguimos optimizar un poco esta búsqueda y bajamos el tiempo hasta un máximo de 2 min si no encontraba ninguna palabra. Esto no es suficiente, ya que haciendo cálculos, el código tardaría en ejecutarse alrededor de 1666 días. Por ello, nos planteamos otras alternativas.

En principio planteamos la función sacar_encadenacion_eligiendo_ganadoras(), la cual miraba si la palabra que matcheaba con la tuya era ganadora. Para ello sacaba inicialmente una lista con todos los matcheos, y mas tarde buscaba entre ellos un ganador. Su tiempo de ejecuión seguía siendo altísimo. Hicimos una prueba con la palabra "mandoble" y tardó 11 min en dar una respuesta.

Como no era viable usar esta función, creamos sacar_encadenacion_eligiendo_ganadora2() la cual confirma si tu palabra es ganadora al obtenerla en el 1º matcheo. Esto te ahorraba recorrer todo el diccionario en primera instancia, pero el coste de ejecuión seguía siendo altísimo y apenas había cambios con respecto al anterior.

En este punto se nos ocurrió implementar un sistema que dividía el diccionario en distintas listas según la letra por la que empezara la palabra. De esta forma el código solo tenía que buscar la palabra en la lista que le correspondiera. Aunque tardan unos minutos en crearse las listas, este trabajo previo permitió reducir el tiempo de ejecución de alrededor de 15 min a unos cuantos segundos, ya que no debiamos recorrer las 1250000 entradas de nuestro diccionario. En resumen, esta función cogerá tu palabra y mirará a que lista corresponde su última sílaba. Una vez encuentre la correspondencia, el programa únicamente buscará dentro de esa lista, ya que no se va a encontrar otra palabra que matchee en otra. Una vez encontramos una coincidencia, usaremos la función auxiliar sacar_cadena_ganadora() para, de nuevo, filtrar para ver en que lista debemos buscar y ver si esa palabra tiene alguna coincidencia. Si la tiene, la palabra elegida no es ganadora y se pasará a la siguiente. Si no encuentra coincidencia, eso significa que la palabra es ganadora y la retornamos. En el caso de que ninguna palabra sea ganadora, se retornará una palabra aleatoria de la lista de matcheos.

Algunos casos excepcionales son las palabras como "gonorrea", ya que el programa marca que su última sílaba es "rrea". Como ya sabemos, no hay ninguna palabra que empiece por rrea en el español, ya que la doble "r" no se puede usar al comienzo de una palabra. Por tanto, si la última sílaba de la palabra del usuario comienza por doble "r", el programa dirá que no existe ninguna palabra que la continue.

Por último, el usuario solo va a poder meter palabras con 5 o más letras que estén en el diccionario. En el caso de que esto no se cumpla se pedirá otra palabra.