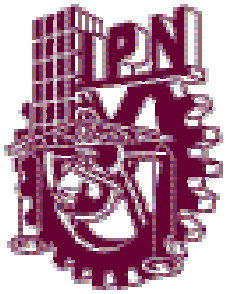


INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
LABORATORIO DE LENGUAJE NATURAL Y PROCESAMIENTO DE TEXTO

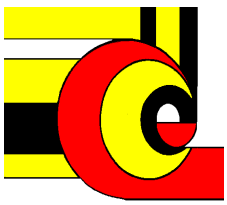
Sistema de Segmentación Automática de Palabras en Morfemas para el Español

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
DIEGO LARA REYES

DIRECTORES DE TESIS:
DR. GRIGORI SIDOROV
DR. ALEXANDER GELBUKH



MÉXICO, D. F. **Abril 2008**

SIP-14

INSTITUTO POLITECNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 17:00 horas del día 28 del mes de Mayo de 2008 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"SISTEMA DE SEGMENTACIÓN AUTOMÁTICA DE PALABRAS EN MORFEMAS PARA EL ESPAÑOL"

LARA	REYES	DIEGO
Apellido paterno	materno	nombre(s)

Con registro: B 0 5 0 9 3 4

aspirante al grado de: ***MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Igor Bolshakov

Secretario

Dra. Nareli Cruz Cortés

Primer vocal
(Director de tesis)

Dr. Grigori Sidorov

Segundo vocal

Dr. Alexander Gelbukh

Tercer Vocal

Dr. Alfonso Medina Urrea

EL PRESIDENTE DEL COLEGIO

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESION DE DERECHOS

En la Ciudad de Mexico, DF el día 06 del mes de Junio del año 2008, el (la) que suscribe Diego Lara Reyes alumno (a) del Programa de Maestria en Ciencias con número de registro b050934, adscrito a Centro de Investigacion en Computacion, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Grigori Sidorov y cede los derechos del trabajo intitulado Sistema de Segmentacion Automatica de Palabras en morfemas para el Español al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección Sidorov@cic.ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Diego Lara Reyes Diego Lara R.

Nombre y firma

Dedicatorias

A mis padres; por su apoyo durante el tiempo que hemos pasado sin compartir.

A la familia que tuve derecho a elegir; por demostrarnos que no necesitamos lazos sanguíneos que nos aten.

Agradecimientos

Al **Instituto Politécnico Nacional** (IPN) por permitirme pertenecer a tan distinguida institución.

Al **Centro de Investigación en Computación** (CIC) por la oportunidad otorgada al creer en mí.

Al **Consejo Nacional de Ciencia y Tecnología** (CONACyT), la **Comisión de Operación y Fomento de Actividades Académicas** del IPN (COFAA) y la **Coordinación de General de Posgrado e Investigación** (CGPI) por el apoyo económico brindado durante mis estudios.

Al **Dr. Grigori Sidorov** por su apoyo, motivación, consejos y enseñanzas durante la realización de este trabajo y durante mi estadía en mis estudios de maestría en general.

Al **Dr. Alexander Gelbukh** por colaborar en la realización de este trabajo y permitirnos el uso su trabajo, herramienta básica para la realización de esta tesis.

Al **Dr. Igor Bolshakov Mironova** por sus enseñanzas y apoyo en la realización de esta tesis.

Al **Dr. Jorge Adalberto Navarro Castillo**, la **M. en C. Imelda Mendivil Angulo** y al **Dr. Cornelio Yáñez Márquez** por ayudarme a mantener fijos mis objetivos.

A mis sinodales **Nareli Cruz-Cortés** y **Alfonso Medina Urrea** por sus atinadas observaciones y propuestas para el mejoramiento de este trabajo.

Índice

Resumen

La segmentación morfológica es un problema que ha desarrollado mayor avance en los últimos años dentro del procesamiento de texto, se han creado varios modelos para esta tarea, modelos tanto supervisados como no supervisados. En este trabajo de investigación se describe un sistema computacional que presenta, hasta cierto nivel, una solución a este problema. Se presenta un método capaz de detectar los conjuntos mínimos de prefijos, raíces y sufijos de las palabras dadas capaces de generar una palabra existente dentro de este conjunto. Se usó el algoritmo genético para esta detección. Se utilizó el concepto de los paradigmas para reducir el conjunto de los posibles afijos.

Abstract

Morphological segmentation is a text processing problem that had significative advances in recent years. Many models have been created for this issue, based on either supervised or non-supervised approaches. In this thesis, a computational system is described, obtaining a solution to this problem to some extent. The method is presented, capable of detecting three minimal sets: prefixes, stems and suffixes of the given set of words, capable to generate already existing words from this group. It is implemented using genetic algorithm. We used the idea of derivational paradigms for reduction the space of possible solutions.

Capítulo 1. Introducción

Usualmente, las palabras son consideradas como la unidad básica de cualquier lenguaje. Las palabras son el modo de comunicación en un lenguaje y varían en tiempo y su uso varia dependiendo del tiempo y del lugar, por ejemplo, existen palabras en inglés que se usan en Inglaterra mientras que las mismas palabras no se utilizan en Estados Unidos de América o viceversa, de manera similar en el español, existen palabras usadas de hace cientos de años que actualmente ni siquiera existen. Otro hecho curioso es que el número de palabras en un lenguaje no está sujeto a un número determinado, de acuerdo a *la hipótesis de separación* de [ALPHADICTIONARY. 06]: "*No existen tales cosas como las palabras, lo que tomamos como palabras son, de hecho, 2 distintos fenómenos de lexemas y morfemas, los lexemas son sustantivos, verbos y adjetivos… los morfemas son todo lo demás, incluyendo los prefijos auto, in, un, intra, ex…*", los lexemas son referencias a cosas del mundo real, mientras que los morfemas son referencias a sus categorías gramaticales.

Un morfema es la más pequeña unidad de una palabra con significado gramatical en un lenguaje. Existen, básicamente, 2 tipos de morfemas; raíces y afijos, las raíz es aquella parte principal de una palabras que se repite una sola vez en la palabra, mientras que los afijos son la parte de la palabras que pueden o no aparecer en ella, los afijos pueden preceder o anteceder, o ambas, a la raíz.

Para poder trabajar con varias partes de las palabras —morfemas— se usa la segmentación morfológica. El objetivo de la segmentación morfológica es el de separar palabras en las unidades más pequeñas provistas de significado llamadas *morfemas*. Cualquier palabra puede ser expresada como una combinación de morfemas, como por ejemplo, las siguientes palabras en inglés:

Mathematicians = Mathematic + ian + s
Arrangements = Arrange + ment + s

O las palabras en español:

Zapatero = Zapat + er + o
Corredores = Corre + dor + es

La segmentación es un problema muy común en el análisis de datos de varias modalidades como la secuencia de genes, análisis de imágenes, series de

tiempo o segmentación de texto a palabras. La información resultante de la división de palabras en morfemas tiene un valor lingüístico importante y puede utilizarse en sistemas de procesamiento de texto, como búsquedas inteligentes, reconocimiento de voz o traducciones automáticas, donde es importante conocer las relaciones derivativas entre las palabras para una traducción correcta.

Un punto de partida para una efectiva segmentación morfológica automática, es considerar como primicia el hecho de poder catalogar a las palabras en conjuntos de significados comunes que pueden compartir desde su raíz (o tema, como se verá en el capítulo 3 de esta tesis) y de esa manera pueda ser posible identificar la palabra independientemente de la forma gramatical que los afijos que otorguen.

1.1 Justificación

La disponibilidad de una herramienta capaz de detectar varias partes de las palabras (en nuestro caso, las 3 partes de una palabra: prefijo, raíz y sufijo) permite acelerar procesos con tiempos de respuesta mínimos en diferentes aplicaciones, como búsquedas especializadas a temas específicos o en bases de datos, traducciones automáticas mucho más eficientes, clasificación de palabras en cuanto a significados, ideal para la automatización de resúmenes automáticos ó incluso en sistemas especializados en reconocimiento de voz, permitiendo una comprensión más detallada de la palabra hablada. Además permite observar los comportamientos que desarrollan las palabras a través de las funciones cambiantes en cada una de ellas, es decir, la formación de los paradigmas.

Es decir, la herramienta sería útil para varias tareas de procesamiento automático de lenguaje natural, como para algunas tareas de la lingüística tradicional, como detección de los paradigmas o compilación de algunos tipos de diccionarios específicos.

1.2 Objetivo General

Implementar una técnica no supervisada de inteligencia artificial (algoritmos genéticos) capaz de realizar un tratamiento de búsqueda y calificación de diferentes soluciones para el problema de generar las segmentaciones de palabras, en este caso, vamos a hacer los experimentos para un conjunto de palabras en español trabajando a nivel de la morfología derivativa (sufijos y prefijos, sin tomar en cuenta las flexiones).

Con los resultados finales, y con un conjunto de palabras lo suficientemente extenso se pueden observar los diferentes comportamientos de la palabra que se dan en el español en cuanto a sus flexiones.

1.3 Objetivos específicos

Las divisiones en las palabras a considerar son 3: sufijos, prefijos y raíces, el algoritmo debe ser capaz de determinar que partes posee cada palabra mediante la información recogida de las demás palabras en el conjunto para realizar la segmentación de cada una de ellas.

También se desea que la herramienta sea lo suficientemente robusta para probarla en otros contextos, como diccionarios de palabras, traducciones o incluso en textos escritos en otros idiomas para observar cómo se desarrollan los comportamientos de las flexiones en el idioma sobre el que se trabaja la herramienta.

Los objetivos específicos son:

- Preparar la lista de palabras.
- Implementar un algoritmo genético.
- Definir la representación de los datos para el algoritmo genético (diseño del cromosoma).
    - Experimentar con varias funciones de evaluación (fitness).
    - Desarrollar la interfaz del sistema.
    - Analizar los parámetros aplacibles del algoritmo genético.
    - Aplicar el algoritmo genético al conjunto de palabras.
- Hacer evaluación preliminar de los resultados comparando con algún sistema existente.

1.4 Importancia

Este trabajo de investigación pretende diseñar una herramienta capaz de generar las divisiones de palabras en raíces y afijos (sufijos y prefijos), tomando todos los afijos que genere el método después de eliminar las flexiones, como se explica en el capítulo 3 de esta tesis.

El problema de división de palabras es importante para las tareas de recuperación de información y para el procesamiento de lenguajes desconocidos.

En este trabajo se propone una variante del trabajo realizado en [GELBUKH, 04], cuyo trabajo consiste en detectar las flexiones en las palabras detectando las raíces de éstas e identificando después de su separación el conjunto mínimo de flexiones resultantes, un análisis más detallado sobre este trabajo se puede encontrar al final del capítulo 2 de esta tesis. Como variante a la investigación descrita en [GELBUKH, 04], este trabajo no solo detecta el mínimo de las flexiones que se dan en las palabras, sino que además hace lo mismo con las raíces y sufijos de estas.

1.5 Estructura de la tesis

Los siguientes capítulos de esta tesis se organizan de la siguiente manera:

Capitulo 2, Antecedentes: nos da una explicación acerca de la herramienta utilizada para encontrar las soluciones (algoritmos genéticos). Explica que es la segmentación morfológica sobre la que se basa el método y una explicación de los trabajos anteriores más importantes que también se desarrollaron para esta misma tarea

Capitulo 3, Fundamentos de morfología: expone las generalidades de la lingüística, la lingüística computacional y se centra en el nivel lingüístico sobre el que se trabaja esta tesis, el nivel morfológico.

Capitulo 4, Construcción del sistema: hace una descripción funcional del método para lograr la segmentación, así como detallar los algoritmos realizados para el desarrollo tanto del algoritmo genético como de la segmentación de palabras.

Capitulo 5, Pruebas y resultados: expone y explica los principales resultados de los experimentos realizados a diferentes conjuntos de palabras.

Capitulo 6, Conclusiones: se muestran las conclusiones a las que se llegaron al final del trabajo y las áreas y trabajos de interés para el seguimiento de esta investigación.

Se cuenta además con una bibliografía en la que se encuentran las referencias a las que se hace durante el desarrollo de esta tesis, conteniendo además los trabajos más relevantes dentro de esta área de investigación.

Capítulo 2. Antecedentes

2.1 Algoritmos genéticos

Los algoritmos genéticos son una herramienta usada para la generación y búsqueda de soluciones en problemas enfocados a la optimización de estas. Su metodología está basada en los mecanismos básicos de la evolución (adaptación, selección, reproducción reemplazo y mutación) para el mejoramiento de las soluciones a través de la ejecución del algoritmo.

Un algoritmo genético tradicional consta de un conjunto de soluciones (generadas por este al inicio) que son representadas en variables y operadores genéticos que se aplicaran sobre estas. Las variables son tratadas por los operadores genéticos durante su ejecución y termina cuando una condición impuesta se cumple.

A continuación se detalla cada uno de los puntos que debe cubrir un algoritmo genético tradicional.

Representación

Usualmente la representación se hace mediante un arreglo unidimensional, esto para facilitar la tarea de cruza. Cada posición del arreglo guarda información codificada de la solución que mediante la decodificación de esta logra calificar cada una como más o menos adaptable al ambiente en el que se encuentra.

Función de adaptación

Es la función principal del algoritmo, se encarga de definir la adaptabilidad de cada solución respecto a las otras, esta función es dependiente del problema sobre el que se trabaja y gracias a esta se definen las soluciones sobre las que se aplicaran los operadores genéticos.

Inicialización

Usualmente, el conjunto inicial de soluciones se genera aleatoriamente para contar con una variedad genética lo suficientemente diversa para la exploración del área de soluciones.

Selección

En cada generación, un determinado número de soluciones son elegidas para generar nuevas a partir de estas, las soluciones a elegir para esta tarea son,

en gran parte, resultado de la calificación que les otorga la función de adaptación, mientras la función de adaptación mejore la calificación de cada soluciones, esta solución tiene más probabilidades de ser seleccionada para generar a las siguientes.

Reproducción

Una vez que se eligieron las soluciones que generaran a las de la siguiente generación, el siguiente paso es el de generar estas nuevas soluciones. Mediante este operador genético se elige una pareja de "padres" que darán, como resultado de su cruza, nuevas soluciones "hijos" que comparten información de los padres. Cada par de nuevos padres se seleccionan para generar tantos hijos como se necesiten para que el nuevo conjunto de soluciones se genere.

Mutación

Una vez en cada generación puede ocurrir que una solución cambie el estado de la información que heredo de sus padres, cambiando aleatoriamente su información por alguna otra, generando una búsqueda (que puede convenir o no) más amplia en el espacio de soluciones.

Finalización

El proceso con los pasos de selección a la mutación se repiten hasta que se cumple una condición de finalización, usualmente las condiciones de finalización para un algoritmo genético son el crear un cierto número de generaciones ó que alguna solución de las creadas cumple con un criterio mínimo de calificación.

2.2 Sistemas existentes

Como se explica en [GELBUKH, 04], existen 3 enfoques principales para la segmentación de palabras; las basadas en diccionarios, en reglas y en métodos estadísticos y probabilísticos.

2.2.1 Enfoque basado en diccionarios

Este enfoque se centra en realizar una recolección de las palabras de un determinado lenguaje junto con sus clases de afijos (tanto prefijos como sufijos) e información extra para poder generar todas las formas morfológicas de una palabra. Teóricamente, una ventaja del uso de este enfoque es que le da un tratamiento correcto a las palabras que parecen tener afijos, aunque estas no lo

tengan, por ejemplo el sustantivo *pintor* lo trata como una variante del verbo *pintar*.

La principal desventaja de este enfoque es que las palabras en el diccionario deben ser constantemente tratadas, la necesidad de procesar, mantener y desarrollar un diccionario con un número tan grande de palabras con un análisis tan complejo es una gran desventaja para este tipo de enfoques.

2.2.2 Enfoque basado en reglas

Para realizar la segmentación de palabras, se siguen una serie de pasos construidos con antelación para tomar las subcadenas que se consideran como afijos o raíces y eliminar estas de la o las palabras para, por medio de estos pasos, ir creando los conjuntos que tomaremos como afijos o raíces. Una desventaja de este tipo de métodos es que las reglas creadas para la segmentación están fuertemente ligadas con el lenguaje sobre el que se quieran utilizar. La realización de las expresiones para la segmentación requiere de un análisis muy detallado de las propiedades lingüísticas del lenguaje sobre el que se desee trabajar.

2.2.3 Enfoque basado en métodos estadísticos y probabilísticos

Este enfoque permite un desarrollo más rápido y totalmente automático para la creación de segmentadores que trabajen en casi cualquier lenguaje.

Muchas aplicaciones de este tipo usan técnicas de aprendizaje supervisado, comenzando a "enseñar" a estos métodos con ejemplos preparados para su posterior "independencia" de supervisión. Sin embargo la sola elección de la colección de ejemplos para el entrenamiento de estos métodos supone toda una problemática para considerar el número de flexiones que existen para cada lenguaje, esto debido a que si se omiten ciertos ejemplos de determinadas flexiones, puede provocar que la segmentación obtenga resultados no deseados u omisos.

Otras aplicaciones, como la mostrada en esta tesis, usan técnicas de aprendizaje no supervisado, por tanto, no es necesaria la intervención de expertos en lenguajes para determinar la validez de la segmentación, puesto que la técnica adquiere su aprendizaje con respecto a la información que se le proporciona, además que se pueden mostrar aspectos acerca de la naturaleza de los lenguajes por medio de textos sin que exista intervención de personas intermedias.

2.2.4 Aplicaciones basadas en los distintos enfoques

2.2.4.1 Segmentación no supervisada en morfemas

En [REHMAN, 00] se crea un modelo (*aprendizaje*) a partir de un corpus dado y prepara una lista de posibles segmentos (morfos) basada en la frecuencia de estos. Después de que el *aprendizaje* esta completo, las palabras son, una por una, segmentadas en base a los posibles morfemas.

El modelo creado en [REHMAN, 00] toma las palabras más frecuentes de un corpus para identificar de estas los posibles morfemas, la frecuencia que debe cumplir una palabra para ser considerada a la segmentación es de siete, aunque, como se ha demostrado experimentalmente, si se toman palabras con una frecuencia menor, el resultado de la segmentación no cambia sino que solamente se vuelve más lento.

Los segmentos a tratar van desde el primer carácter de cada palabra hasta el decimotercero, ya sea de izquierda a derecha (de encabezado) de la palabra o de derecha a izquierda (de seguimiento). La formación del conjunto de posibles morfos se realiza de la siguiente manera:

1. Comenzando con la cadena vacía, a partir del primer carácter y para cada palabra, se toma el carácter y se concatena con la subcadena actual.
2. Con la subcadena con la que se cuenta, se verifica la frecuencia con la que se encuentra al inicio (de izquierda a derecha) de cada palabra.
3. Si la subcadena existe como una palabra, entonces esta subcadena es calificada como un segmento valido y agregada a una lista de *segmentos aprendidos*.
4. Se continua de nuevo a partir del paso 1, sustituyendo la cadena vacía por la subcadena con la que se cuenta hasta el carácter decimotercero de la palabra (si la longitud de la palabra es mayor a trece), si se cumplen todos los caracteres de la palabra o se llega al decimotercero, entonces se continua el proceso con la siguiente palabra hasta cumplir con todas las palabras de la lista.

De manera similar, se realiza el mismo proceso para tomar segmentos de las palabras, solo que de derecha a izquierda, comenzando con el último carácter de cada palabra.

En caso de que un segmento similar sea encontrado en ambos procesos, se verifica la frecuencia con la que se repite el segmento (tanto de izquierda a

derecha de cada palabra, como de derecha a izquierda) y se agrega a la lista de *segmentos aprendidos*.

Si durante el proceso de formación de *segmentos aprendidos* entran caracteres que sean consonantes y de longitud igual a uno, entonces estos caracteres se consideraran como no validos. Para ignorar estos caracteres se calculan sus *pesos*, esto se hace restando la desviación estándar de frecuencias con longitud uno a las frecuencias de cada uno de estos segmentos. Si el peso resultante del carácter es negativo, entonces el carácter no es considerado como un segmento valido.

Una vez con la lista de *segmentos aprendidos* "limpia" se comienza la segmentación en dos fases. La primera fase verifica las subcadenas que forma cada palabra del corpus sobre el cual se lleno la lista desde el primero hasta el último carácter, si la subcadena a analizar se encuentra dentro de la lista de *segmentos aprendidos*, entonces esta subcadena es removida de la palabra y si la subcadena que le resta también está dentro de la lista, entonces el segmento removido de la palabra es considerado como un posible prefijo. En caso de que el segmento removido de la palabra sea de dos o menos caracteres, entonces se calcula su *peso*, si resulta ser negativo, entonces el segmento no se considera como prefijo. Este proceso continua hasta que se encuentre un prefijo valido. La cadena restante (después de removerle el prefijo) pasa a un modulo de separación de sufijo, este modulo comienza desde un primer carácter de seguimiento y continua hasta una segmentación máxima de tres caracteres de seguimiento, lo que puede llevar a generar más de un sufijo, en este caso, el sufijo con mayor *peso* es considerado como el valido.

En caso de que la separación del prefijo de la palabra falle, la palabra no pasa al modulo de separación de sufijo, pasa a la segunda fase del modelo que consiste en separar a la palabra desde el primer carácter de seguimiento y pasar la cadena restante al modulo de separación de prefijos. En este punto, el modulo de separación de prefijos realiza el mismo proceso que en la primera fase, solo que con un carácter menos. Esta segunda fase repite el proceso cortando uno a uno los caracteres de seguimiento y enviando las cadenas restantes al modulo de separación de prefijos hasta que el modulo encuentra un prefijo valido. Esta segunda fase es útil para aquellas palabras que no son posibles de separar durante la ejecución del modulo de separación de prefijos, es de ayuda para el tratamiento de palabras cuyos -dos- segmentos validos no se encuentran dentro de la lista de *segmentos aprendidos*.

2.2.4.2 Segmentación no supervisada en morfemas usando la distribución basada en longitud y frecuencias de morfos

[CREUTZ, 03] desarrolla una aplicación basada en el enfoque probabilista para segmentar palabras en lenguajes aglutinativos, específicamente el finlandés. Comienza calculando los posibles morfos que existen en un determinado número de palabras y seleccionando aquellos que cumplen con las condiciones que imponen con el lenguaje seleccionado, como la longitud de las cadenas de cada morfo, la cantidad de veces que se repite cada afijo dentro del conjunto de palabras y la frecuencia con la que cada morfo se repite dentro del conjunto de palabras sobre las que trabaja la aplicación. Cada uno de estos parámetros son ajustados por el usuario para adaptarlos al lenguaje sobre el que trabaja la aplicación, por lo que [CREUTZ, 03] puede considerarse un método basado en métodos estadísticos y probabilísticos, pero fuertemente respaldado por el enfoque que se basa en reglas.

2.2.4.3 Detección de patrones de flexiones usando minimización del modelo morfológico

En el método propuesto en [GELBUKH, 04] se genera una lista de posibles raíces y sufijos a partir de una lista de palabras lo suficientemente grande dadas en cualquier lenguaje. Puede descomponer cualquier palabra en 2 partes: raíz y sufijo (o terminación, o flexión de la palabra) a partir de los conjuntos que el método forma, en caso de ambigüedad se selecciona la raíz y el sufijo más frecuentes determinados por el método. También véase un enfoque parecido de [KAZAKOV, 97].

El método es guiado a partir de 2 hipótesis:

1. Las palabras del lenguaje no son más que simples concatenaciones de una raíz y un –posiblemente complejo- sufijo (o prefijo). Por lo que se ignora el fenómeno Sandhi (por ejemplo, el que se presenta en la palabra *bloc* por su plural *bloc* + *es* = *bloques*) y otras complicaciones morfológicas del lenguaje en la realidad.
2. El lenguaje está formado de tal manera que la forma de aprenderlo requiere del mínimo esfuerzo, por lo que se requiere conocer de un mínimo numero de raíces y sufijos, por lo que estos conjuntos de subcadenas pueden ser reutilizados para formar combinaciones de palabras, por ejemplo: *salt-o*, *salt-ar*, *salt-an*, *dibuj-o*, *dibuj-ar*, *dibuj-an*.

Básicamente, se crean 2 conjuntos de subcadenas a partir de una lista de palabras, el conjunto *S* (*raíces*) y *E* (*prefijos*). Se considera cada palabra (*w*) como

una concatenación de un elemento de S con un elemento de E, esto es: $w = s + e$, $s \in S$, $e \in E$ y $|S| + |E|$ tenga el valor mínimo sobre todos los conjuntos que cumplan con tales características, donde $|X|$ es el número de elementos en el conjunto X. Se buscan los conjuntos mínimos de S y E que generen a V, considerando que V es un subconjunto de $S + E$.

Para encontrar los conjuntos S y E se utiliza un algoritmo genético y, como se menciona en [GELBUKH, 04], cualquier otro método de optimización puede ser usado. La longitud de los cromosomas es de $|V|$ y cada gen representa el punto de división de cada palabra de V. Para cada posible solución que genera el algoritmo se calcula el número total de raíces $|S|$ y prefijos $|E|$, y para hacer notoria la preferencia de un número menor de prefijos por debajo del número de raíces se considera la siguiente función de evaluación:

$$|S| + 0.00000|E| \rightarrow \min$$

esto solo afecta los casos cuando los cromosomas tienen el mismo número de raíces y sufijos.

Puesto que el algoritmo trabaja mejor cuando el conjunto de palabras es muy grande, en ese caso, el espacio de búsqueda los es más aun, para reducir este espacio se consideran cromosomas con valores binarios indicando la presencia con un 1, o ausencia con un 0 de las posibles raíces y prefijos en los conjuntos S y E. Para esto, se generan los conjuntos S' y E' de todas las posibles raíces y prefijos existentes en V. De ellos, se eliminan todos aquellos elementos que solamente se repiten una vez, si una descomposición de una palabra $w \in V$ en $w = s + e$ y e se encuentra dentro del conjunto una sola vez, la descomposición queda en $w = w + \lambda$, donde λ es una cadena vacía.

Cuando un elemento es eliminado de E' la frecuencia del elemento que forma la palabra ($s \in S'$) puede decrementarse también, incluso hasta que su frecuencia sea igual a uno, en ese caso, el elemento s, es también eliminado de S', esa eliminación reduce los conjuntos S' y E', formando así los conjuntos S'' y E'', asegurando así que cada elemento s $\in S''$ tiene al menos dos diferentes $e \in E''$ con los que puede formar una palabra valida que este dentro de V.

Con esto, se forma un cromosoma binario con longitud $|S''| + |E''|$ con soporte para la inclusión de un elemento de los conjuntos S ó E en cada gen con valor igual a 1. Si para una selección de S o E, una palabra $w \in V$ no puede ser dividida, se considerará esa palabra w como un nuevo elemento de S, por lo que en [GELBUKH, 04] se genera una nueva función de evaluación:

$$|S| + 0.000001\ |E| + |V \setminus (S + E)| \rightarrow \min.$$

La parte agregada de la expresión corresponde a aquellas palabras de V que no pueden ser divididas por el método y que se agregan a S durante la ejecución de este.

Como se verá en un capitulo más adelante de esta tesis, el método de segmentación de palabras propuesto en la presente está fuertemente basada en este modelo, con la clara diferencia de agregados en las hipótesis a partir de las que se basa este modelo, y por esta misma razón, las formas de evaluar cada una de las posibles soluciones que se presentan.

Capítulo 3. Fundamentos de morfología

Una característica de los seres humanos esencial y distintiva para con los demás seres vivos es su capacidad para comunicarse a través de la lengua, la cual surge por la necesidad de escuchar y ser escuchar, de interrelacionarse.

Según [NIDA, 86] existen 8 maneras de usar el lenguaje:

1. Estética: como en la poesía o la publicidad.
2. Cognitiva: el uso de las palabras al pensar.
3. Emotiva: el uso de las palabras para provocar sentimientos en los receptores.
4. Expresiva: usada para demostrar las emociones al exterior.
5. Informativa: para el intercambio de información.
6. Imperativa: usada para influenciar el comportamiento de los demás.
7. Interpersonal: mantiene relaciones, además de formarlas.
8. Formativa: modifica el estado de los receptores.

Además de las 8 maneras mencionadas, se usa también de forma documental e interrogativa para almacenar y obtener -respectivamente- información.

La lingüística computacional trata el estudio científico del lenguaje desde una perspectiva computacional, proporcionando modelos computacionales diferentes para cada tipo de comportamiento o fenómeno lingüístico. Dentro de la Lingüística computacional existen muchos campos de estudio, uno de ellos es el de la morfología computacional, que sirve para el desarrollo de sistemas para el análisis y síntesis automática al nivel morfológico, de igual manera el tratamiento de las palabras a nivel morfológico, nos permite descifrar las diferentes partes de la palabra que contienen significado, además de crear reglas que nos permiten observar y clarificar las diferentes ramificaciones que puede tomar cada palabra. Existen además otras áreas de estudio dentro de la lingüística como la fonología, la fonética, la sintaxis, la pragmática y el discurso, cada una de ellas con un campo de estudio muy bien definido.

De acuerdo a lo anterior, los niveles que tradicionalmente trata el estudio del lenguaje natural son los siguientes:

- La fonética y la fonología.

- La morfología.
- La sintaxis.
- La semántica.
- La pragmática.
- El discurso.

Cada estudio de cada nivel del lenguaje se diferencia de otro según las distintas entidades lingüísticas en las que analiza cada uno de los niveles.

Nosotros vamos a trabajar en el nivel morfológico.

El nivel morfológico del lenguaje consta de una serie de unidades de distinto rango, como la palabra, base, tema y morfema. La *palabra*, unidad de rango mayor y objeto de estudio de la morfología presenta una serie de propiedades formales específicas que atañen a su constitución.

Como objeto de estudio, la morfología tiene la estructura interna de la palabra, y sus objetivos son:

- Delimitar, definir y clasificar las unidades del componente morfológico.
- Describir como estas unidades se agrupan en sus distintos paradigmas; los paradigmas son modelos que funcionan como reglas para conjugaciones y declinaciones de la palabra.
- Explicar el modo en el que las unidades integrantes de la palabra se combinan y constituyen, conformando así su estructura interna.

3.1 Partes de la morfología

El estudio de la morfología es comprendido por 2 partes, la *morfología flexiva* y la *morfología léxica*, esta división corresponde con los tipos de palabras establecidos según los la naturaleza de los morfemas que la integran y la estructura que configuran tales morfemas como elementos constitutivos de las palabras. En el español existen 3 tipos de palabras que interesan destacar:

1. Monomorfémicas o Polimorfémicas.
2. Variables o flexivas.
3. Simples y complejas.

La primera clasificación permite distinguir las entre las palabras en su versión reducida, constituidas por un solo morfema, y las palabras integradas por

más de un morfema y por ello con estructura interna, por ejemplo las palabras *ayer* frente a *blanc-o-s*.

La segunda clasificación hace referencia al hecho de que una misma palabra pueda variar formalmente o no según las construcciones sintácticas de la que forme parte, por ejemplo *blanc-o*,*-a*,*-o-s*,*-a-s* frente a *ayer* o *anteayer*.

La tercera alude a palabras diferentes en cuanto a su estructura, pero relacionadas formal y semánticamente, por ejemplo *ayer* y *blanco* frente a *anteayer*, *blancuzco* y *blanquinegro*.

Desde el punto de vista morfológico, la justificación para afirmar que las palabras *blanco* y *blanca* son dos formas de la misma palabra mientras que *blanco* y *blancuzco* son dos formas de palabras distintas se basa en la noción de **tema**, que definiremos provisionalmente como *la unidad constante o abstracta que resulta de eliminar en la palabra los morfemas flexivos*. En las palabras *blanco* y *blancuzco*, sus temas correspondientes son *blanc-* y *blancuzc-*, respectivamente.

El tema como entidad abstracta, es una unidad virtual que no se manifiesta como tal en las construcciones sintácticas, sino como palabra flexiva. Se trata de una unidad necesaria para el análisis morfológico, cuya naturaleza y estructura resultan sintácticamente irrelevantes.

A su vez también, existen dos tipos de morfologías que actúan sobre las palabras, la *morfología léxica*, que se encarga de la formación de nuevas palabras, si tomamos la unidad tema tal como acabamos de definirla, podemos decir que el objeto de estudio de la morfología léxica es el análisis de los temas complejos de las palabras ya existentes y la formación de temas de nuevas palabras, tomando por ejemplo la palabra *pintor*, sustantivo relacionado al verbo *pintar* que significa "persona que pinta", en este caso es la terminación *–or* a partir del tema *pint-* la que conduce el significado y crea un sustantivo a partir de un verbo, ya que *pintor* no es una forma del verbo *pintar*, tal como lo es *pintando* o *pintaba*, cuyos temas y significados son los mismos. El otro tipo de morfología es la *morfología flexiva*, que se ocupa de las variaciones de una misma palabra, esta morfología tiene como objeto de estudio el análisis o formación de las distintas formas de las palabras construidas sobre el mismo tema, por ejemplo, esta morfología se encarga de formas las palabras *blanc-o*, *blanc-a*, *blanc-os*, etc.

3.2 Unidades del análisis morfológico

Uno de los objetivos de la morfología es el de delimitar las unidades con las que opera el análisis morfológico y agruparlas en tipos y subtipos cuyos integrantes comparten determinadas propiedades. Las unidades básicas del componente morfológico son; la *palabra*, el *tema*, y el *morfema*.

3.2.1 Introducción a la palabra y morfema

La palabra, como toda unidad compleja puede ser analizada en unidades menores, que son sus elementos integrantes. Analizar una palabra es descomponerla en sus constituyentes inmediatos en sucesivas etapas hasta llegar a delimitar las unidades gramaticales mininas, denominadas *morfemas*. Así en el análisis de la palabra *trabajadores*, obtenemos los morfemas *trabaj-a-do-r-es*. Los morfemas están representados por segmentos fonéticos o significantes denominados *morfos*. Un morfema puede estar representado siempre bajo la misma forma fonética o morfo ó bajo distintas formas fonéticas o *alomorfos*, un ejemplo claro de alomorfos son las terminaciones *–s* y *–es* para distinguir las formas plurales de *gato-s* y *león-es*.

La palabra y el morfema son dos unidades imprescindibles en el análisis morfológico: la palabra, como unidad de rango superior objeto de estudio de la morfología y el morfema, como constituyente ultimo de la palabra o unidad gramatical mínima. Ambas unidades son unidades morfológicas necesarias en el análisis, pero no suficientes.

3.2.2 Raíz y afijos

Después de las breves consideraciones sobre la estructura de la palabra, pasemos a describir las unidades raíz y afijos, pertinentes en el análisis de la palabra tomando una serie de palabras de la misma familia léxica en estas distintas formas flexivas:

blanc-o, -a, -o-s, -a-s
blancuzc-o, -a, -o-s, -a-s
blancot-e, -a, -e-s, -a-s
blancaz-o-, -a, -o-s,-a-s
blancura, -s

Si las analizamos desde los morfemas que las integran, se observa que todas tienen un significante en común, este es representado, en este caso, por /blanc/. Se trata del segmento básico y constante en el significante de cualquier palabra que, como resultado de eliminar todos lo afijos derivativos y /o flexivos, es irreductible y no permitible a mas análisis. De esta manera se hace una distinción en este significante que es irreductible y los significantes que se le adjuntan… al primero se le llama *raíz*, los demás son *afijos*.

Si las palabras analizadas anteriormente son observadas más detalladamente, al punto de notar los afijos, notamos que hay unos que se adjuntan directa o indirectamente a la raíz y que constituyen con ella el tema de las distintas clases de palabras, como *–uzc* en *blanc-uzc-o*, o *–ot* en *blanc-ot-e* y otros que se adjuntan al tema ya constituido y lo adaptan para la expresión de las categorías gramaticales que cada clase de palabras flexivas soporta, por ejemplo los morfos *-o-* de masculino y la *–s* de plural. Según que los afijos formen parte del tema o se adjunten a él se habla de *afijos derivativos* y de *afijos flexivos* respectivamente.

Los afijos derivativos forman parte del tema y sirven para crear palabras relacionadas semánticamente. Los afijos flexivos se adjuntan externamente al tema y crean diferentes formas de la misma palabra, que sirven para expresar las distintas propiedades o categorías gramaticales exigidas en las construcciones sintácticas.

El conjunto de palabras formadas con afijos flexivos sobre un mismo tema se integra en un conjunto cerrado denominado paradigma flexivo. Por ejemplo, el paradigma del sustantivo *trabajo* consta de dos formas flexivas, *trabajo* y *trabajos*, mientras que el sustantivo trabajador consta de cuatro formas flexivas, *trabajador*, *trabajadora*, *trabajadores* y *trabajadoras*.

3.2.3 Morfema

El morfema es la unidad mínima del análisis morfológico, la *unidad mínima con significado*, partiendo de la idea de que se trata de la unidad mínima quiere decir que no se puede descomponer en mas unidades, tenemos así, en el análisis morfológico de la palabra *pintores*, lo delimitamos en tres partes *pint-or-es* asociados con los significados *pintor*, con coordenadas gramaticales sustantivo y plural. Estas tres separaciones de la palabra como unidades mínimas dejan de ser analizables por otros signos, lo que si es posible hacer con estas unidades es analizar el significado de un morfema.

3.2.4 Palabra

La *palabra*, no es una unidad general a todas las lenguas, como tampoco son generales las propiedades que la distinguen de las otras unidades. Su existencia y sus propiedades son dependientes de los tipos morfológicos de cada lengua. En el español existe la palabra como unidad propia.

La definición de la palabra, sin duda, necesita de más propiedades para definirla como tal. Por ello existen 3 características de la palabra detectables desde una primera perspectiva:

1. Posibilidad de cambiar su posición en la secuencia, esto es, de mantener distintas relaciones secuenciales con los demás elementos de la oración, por ejemplo en la oración *Él siempre va a casa*, *Él va a casa siempre*, *Siempre va él a casa*, *A casa va siempre él*, etc.
2. La separabilidad: entre 2 palabras es posible insertar otras u otras unidades, por ejemplo *El auto esta en casa*, *El auto nuevo esta en casa*, *El auto nuevo que me esta estorbando esta en casa*.
3. El espacio: cuando se escribe o se pronuncia un enunciado, existe un espacio o una pausa entre cada palabra.

En cuanto a la relación de las palabras con elementos externos, estas son las características de las palabras. Ahora se enumerarán las propiedades relativas en cuanto a su estructura interna, refiriéndonos a las palabras polimorfemicas:

1. El orden fijo de los morfemas no admite una reestructuración, admite solamente distintas relaciones secuenciales en su interior, por ejemplo, en la palabra *niños* la raíz *niñ-* precede al morfo *-o-* y *-o-* precede al morfo *–s*. No es admitible el ordenarlos de otro modo.
2. La palabra no admite más adiciones que la de los morfemas ligados, por ejemplo *blanc-o*, *blanc-uzc-o*, *blanc-ot-e*, etc.
3. La imposibilidad de separación de los morfemas integrales de la palabra; por ejemplo, no es posible extraer las desinencias –ras o el sufijo –miento al decir *cantaré y rás* o *estanca y empobrecimiento*, sin embargo, hay algunos casos de separabilidad permitida en el proceso de derivación, con algunos prefijos, por ejemplo *becas pre- y posdoctorales*, de igual manera ocurre con el sufijo *–mente*, *actúa audaz y rápidamente*.

Cabe observar que las mismas propiedades que marcan positivamente a la palabra en cuanto a constituyente de una unidad superior, la marcan negativamente en cuanto a su estructura interna. En resumen, la palabra, en español por lo menos, se caracteriza por la inseparabilidad y el orden fijo de los morfemas que la integran.

Con lo expuesto anteriormente, la cuestión de que si una unidad es una palabra o no, no puede plantearse en términos de *si o no*, sino en términos de *más o menos* según cumpla un número mayor o menor de las propiedades definitorias de la unidad palabra.

3.3 La segmentación

La segmentación es la primera de tres etapas sucesivas del análisis de la palabra. El analizar morfológicamente una palabra consiste en descomponerla en sus partes constituyentes hasta llegar a delimitar e identificar las unidades gramáticas mínimas. Como ya se mencionó, la segmentación es la primera y, por la naturaleza tratada en esta tesis, la única en la que se interesa profundizar. De manera superficial, las tres etapas son:

1. Segmentar la forma fonética de una palabra en los segmentos fonéticos mínimos que porten un significado constante, denominados, como ya se explico, morfos.
2. Agrupar como alomorfos de un mismo fonema aquellos morfos que expresen un significado.
3. Describir y organizar de manera sistemática aquellas diferencias fonéticas que se repitan entre los alomorfos de dos o más morfemas.

El primer paso, la segmentación, se parte del siguiente postulado: la parte común a dos o más palabras consta de un tema constante asociado a un significado también constante. De lo que se trata es de delimitar e identificar esa unidad en esa unidad compleja aquellos segmentos fonéticos portadores de un determinado significado, que reaparecen en otras unidades (palabras). Para esto hay que comparar y hacer contraste palabras parcialmente iguales y diferentes y proceder mediante segmentación y conmutación de un segmento por otro, hecho un corte o segmentación en la forma fonética de una palabra, hay que conmutar cada uno de los segmentos obtenidos por otro segmento en ese mismo contexto. Si lo que resulta de la conmutación es el significado global de la palabra analizada y si los elementos conmutados reaparecen en otras unidades con el mismo significado que en la unidad analizada, estamos ante dos signos diferentes, y el análisis prosigue hasta obtener los signos mínimos. Por ejemplo, si tomamos como objeto de análisis la forma flexiva de la palabra *cantábamos* podemos probar a segmentarla en *cantá-bamos*: el segmento *cantá-* es conmutable por *contá-* y obtenemos *contábamos*, con un significado parcial diferente; por otro lado, tanto *cantá-* como *contá-* aparecen en otros contextos distintos de *–bamos* dentro del paradigma flexivo: *cantáramos*, *cantamos*, *contáramos*, *contamos*, etc.

Los morfos son, en principio segmentos fonéticos recurrentes con un significado constante, esto es, segmentos que reaparecen en otras unidades con la misma forma fonológica y con el mismo significado. Lo que quiere decir que son elementos que se combinan libremente unos con otros: los segmentos que aparecen en una palabra pueden aparecer separadamente en otras, ahora, la libertad en la combinación admite grados, puede suceder que:

a) Un mismo segmento sea independiente y, por lo tanto, separable y combinable libremente con otros en unos contextos, y parcialmente dependiente y combinable en otros.
b) Que existan segmentos no autónomos y combinables solo con determinados segmentos.
c) Que ocurra el caso extremo en que un segmento no aparece más que en combinación con otro segmento.

El resultado de la primera etapa del análisis es una lista de los morfos de una lengua como queda indicado en principio, los morfos son segmentos recurrentes con el mismo significado. Pero también sucede que segmentos fonéticos parcialmente distintos aparecen expresando el mismo significado; es lo que ocurre, por ejemplo, con el segmento constante *tiran-* en *tirano*, *tiranía* y *tiránico*.

Capítulo 4. Método propuesto

4.1 Método de segmentación propuesto

El proceso de segmentación se desarrolla de la siguiente manera; como entrada, recibe un conjunto de palabras dadas como toda su información disponible y se comienzan a generar las todas posibles divisiones.

1. Se calculan todos los posibles prefijos, raíces y sufijos de cada palabra.
2. Todas estas posibles cadenas se representan como un cromosoma, para ser tratadas dentro de un algoritmo genético tradicional.
3. Se detectan y se filtran los paradigmas que contienen únicamente un elemento (sin contar la cadena vacía) y con frecuencia menor que la establecida por el usuario (ver capítulo 4.3).
4. Aleatoriamente, se genera un conjunto de posibles soluciones (individuos).
5. Cada solución se evalúa dependiendo de la validez que tiene cada concatenación de subcadenas (posible prefijo + posible raíz + posible sufijo) dentro del conjunto de palabras dadas al inicio del algoritmo.
6. La mejor solución es usada para conocer los prefijos, raíces y sufijos válidos para el conjunto de palabras.
7. Cada palabra se divide usando la información de esta solución.

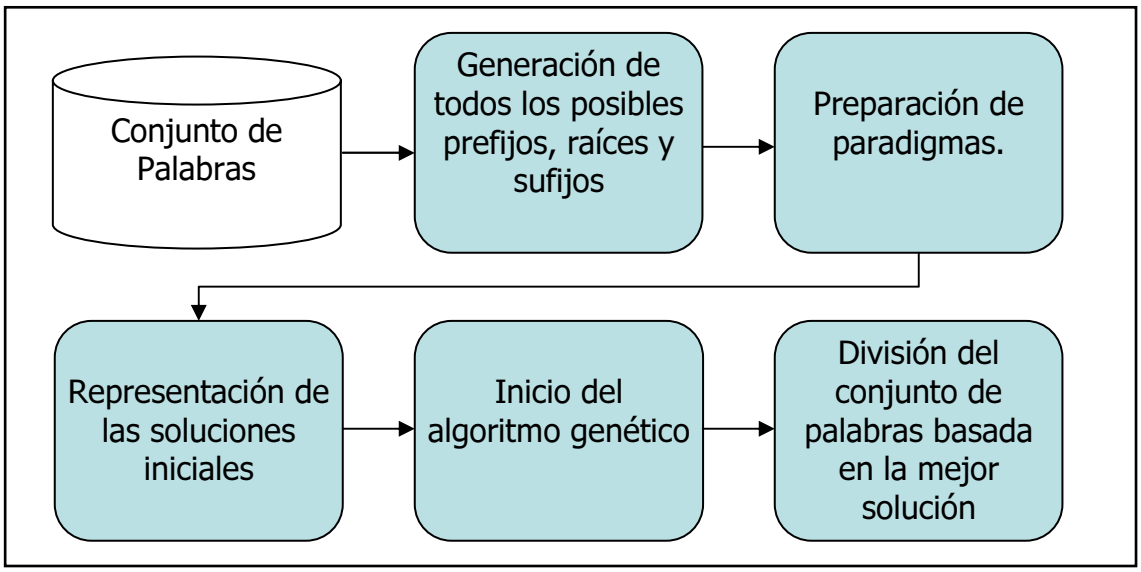


Figura 4.1. Proceso del algoritmo de segmentación

Más detalladamente, dado un conjunto de palabras, se analizan de la siguiente manera:

1. Por cada palabra se calculan todos sus posibles prefijos (contando desde la subcadena ∅) comenzando desde la primera letra y continuando hasta la penúltima letra de cada palabra, sin permitir que se repitan estas subcadenas.
2. Se calculan todas las posibles raíces de cada palabra, a partir de la primera letra hasta la última letra, no se toma en cuenta la subcadena ∅, ni se permite que se repitan las subcadenas candidatas a raíces.
3. Se calculan todos los posibles sufijos de cada palabra (comenzando con la subcadena ∅) tomando letra por letra desde la última letra de cada palabra hasta la segunda letra de cada palabra, sin permitir que se repitan estas subcadenas.
4. Cada subcadena formada (tanto de posibles prefijos como de posibles raíces y posibles sufijos) se almacena para ser tratada dentro de un algoritmo genético.
5. Se detectan los paradigmas y se filtran los paradigmas que contienen únicamente un elemento (pero no el paradigma con el elemento vacio) y los paradigmas con frecuencia menos que tres.
6. Aleatoriamente, se generan las posibles soluciones en forma de un cromosoma binario, asignando a cada subcadena un gen dentro del cromosoma, por lo tanto, el tamaño del cromosoma es igual a la suma de los posibles prefijos, raíces y sufijos, y cada gen dentro del cromosoma con

valor igual a 1 representa a una subcadena válida a tratarse como prefijo, raíz ó sufijo.

7. Se ingresan las soluciones iniciales a un algoritmo genético.
8. Cada solución puede obtener mejor puntuación que las demás basándose en las siguientes condiciones:
    a. El número de subcadenas válidas del cromosoma es menor que el de los demás, esto es para asegurar el menor número posible de subcadenas en las que se pueda dividir cada palabra.
    b. La combinación de subcadenas válidas en la solución (aquellos genes dentro del cromosoma con valor igual a 1) de los 3 tipos conforman palabras válidas dentro del conjunto de palabras dadas al inicio. Se realizan todas las combinaciones entre prefijos, raíces y sufijos.
9. Al terminar el algoritmo genético, la mejor solución es usada para determinar cuales son los prefijos, raíces y sufijos del conjunto de palabras, las palabras se dividen de acuerdo a esta solución.

Es importante hacer notar que la solución mostrada al final de la ejecución del algoritmo es estrictamente basada en la información otorgada por el conjunto de palabras dadas al inicio.

La idea de detección de los patrones repetitivos de afijos —potenciales paradigmas— es muy importante para el método propuesto. Sus resultados a veces se llaman "firma (*signature*)", sin embargo preferimos el término más lingüístico "paradigma", en nuestro caso hablamos de los paradigmas derivacionales. Este concepto se refiere al hecho de que las raíces se pueden concatenar con cierto conjunto de afijos, y estos conjuntos de afijos se repiten en el léxico, por ejemplo, *trabajo, trabajemos, trabajador* y *cobro, cobremos, cobrador* tienen el conjunto de afijos {-o, -emos, -ador}. El uso de los paradigmas reduce en gran medida el espacio de búsqueda del algoritmo.

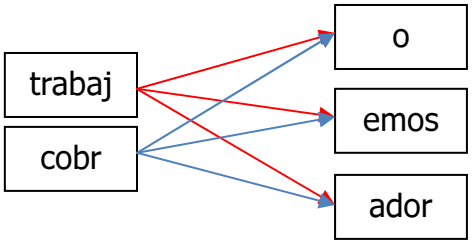


Figura 4.2. 2 paradigmas como raíces y 3 paradigmas como sufijos pueden formar 6 palabras validas

Utilizamos como una función de evaluación la siguiente fórmula:

$$fitness = -\log(freq(x0)) - \log(freq(prefix) * freq(stem) * freq(suffix))$$

donde

freq(x0)) es suma de todas las posiciones en cromosoma donde se encuentran ceros,

freq (prefix) es suma de las frecuencias de todos los prefijos que participan en la solución (su gen es igual a 1),

freq (stem) es suma de las frecuencias de todas las raices que participan en la solución (su gen es igual a 1),

freq (suffix) es suma de las frecuencias de todos los sufijos que participan en la solución (su gen es igual a 1).

Dado que se minimiza el tamaño total de la lista de elementos, se puede considerar esta función como una variedad de la idea de descripción con longitud mínima (minimum length description, MDL) [GOLDSMITH, 01].

4.2 Descripción del método

A continuación se describe la implementación del algoritmo.

Para calcular todas las subcadenas candidatas a ser prefijos, raíces y sufijos se utiliza la función **Segmenta_Palabras**, recibiendo como parámetros un arreglo de cadenas con todo el conjunto de palabras a analizar, el número de palabras que son, un parámetro que devuelve la lista completa con los prefijos, sufijos y raíces, un arreglo que indica la frecuencia con la que aparece cada cadena de la lista de las subcadenas que almacena el arreglo **Lista_de_Cadenas**, 3 parámetros que devuelven el número de posibles prefijos, raíces y sufijos, 3 parámetros con los valores mínimos que requiere cada subcadena para ser tomada en cuenta como candidata a prefijo, raíce y sufijo, 3 parámetros con la longitud mínima que requiere cada subcadena para ser tomada en cuenta y 3 arreglos que devuelven por separado todas las posibles cadenas a ser tomadas en cuenta por el algoritmo genético (ver Anexo 1, función Segmenta_Palabras).

Para el tratamiento y representación de las posibles soluciones del problema dentro del algoritmo genético, se utilizó una estructura llamada **Cromosoma**, conteniendo toda la información necesaria de cada solución a ser tratada por el algoritmo genético, un arreglo de enteros para su representación binaria, un valor

real que nos guarda la evaluación de la solución, la probabilidad que tiene la solución de ser mutada y la probabilidad de la solución para ser reemplazada en una generación posterior (ver Anexo 1, estructura Cromosoma).

La representación de las soluciones a ingresar al algoritmo genético son generadas de la siguiente manera; a manera de ejemplificar esta representación:

Tomando como conjunto dos palabras: *hola*, *adiós*, se genera, como representación un arreglo con todas las subcadenas que generan estas palabras.

Prefijos	Raíces	Sufijos
∅	h	∅
h	ho	a
ho	hol	la
hol	hola	ola
a	o	s
ad	ol	os
adi	ola	ios
adio	l	dios
	la	
	a	
	~~a~~	
	ad	
	adi	
	adio	
	adios	
	d	
	di	
	dio	
	dios	
	i	
	io	
	ios	
	o	
	os	
	s	

Figura 4.3. Subcadenas con los posibles prefijos, raíces y sufijos para las palabras ***hola*** y ***adiós***

Al final del arreglo con los posibles prefijos se concatena el arreglo con las posibles raíces y el arreglo con los posibles sufijos, creando, en este ejemplo, un arreglo con 40 posiciones donde los posibles prefijos abarcan de la posición 1 a la 8 del arreglo, las posibles raíces de la posición 9 a la 32 y los posibles sufijos de la 33 a la 40. La subcadena *a* remarcada entre las raíces no se toma en cuenta porque ya

se había agregado anteriormente en el arreglo, se muestra solo para la comprensión del método de creación de cadenas candidatas para la separación. De esta manera el tamaño del cromosoma será definido por el número de subcadenas que se obtengan del conjunto de palabras.

Un arreglo de la estructura **Pareja** es llenado mediante el operador genético **Selección** y en él se pueden encontrar a los individuos seleccionados para crear un nuevo conjunto de individuos que reemplazaran a aquellos con evaluación menor a los demás (Ver Anexo 1, estructura Pareja)

Para conocer la posición en la que se encuentra cada subcadena y el número de veces que esta fue repetida en el texto a analizar se utiliza una estructura llamada **TFrecYPos** que nos indica en sus valores **frec** y **pos** la frecuencia y la posición de la subcadena respectivamente (ver Anexo 1, estructura TFrecYPos).

Cada uno de los operadores genéticos del algoritmo genético fue representado como una función modular que cumplen solo a su propósito, generar una solución inicial, evaluar cada solución, seleccionar a los mejores individuos a reproducirse, cruzar a los individuos para crear un conjunto de nuevas soluciones, reemplazar a los peores individuos con los recién generados y mutar a los individuos para expandir el campo de búsqueda de soluciones. Pero antes de explicar con detalle cada uno de estos, se hizo uso de 3 funciones de soporte para el funcionamiento de las recién mencionadas.

La función **Dimensiona_Poblacion** solo crea un conjunto de individuos sin ninguna información, listos para ser llenados a partir de cualquiera de los operadores genéticos (ver Anexo 1, función Dimensiona_Poblacion)

Torneo -en este caso- recibe como parámetro a todos los individuos de la población, elige a 2 diferentes individuos, los compara y el mejor individuo es elegido para su posterior cruzamiento (ver Anexo 1, función Torneo)

La función **Ordena_Poblacion** recibe como parámetro a toda la población y la ordena según su evaluación de menor a mayor con el fin de eliminar de la población a los últimos individuos, este número de individuos es decidido según el porcentaje de reemplazo asignado al algoritmo genético (ver Anexo 1, función Ordena_Poblacion)

Los operadores genéticos a mostrar con más detalle son:

1. La generación de la población inicial.
2. Evaluación de la población actual.
3. Selección de individuos
4. Cruzamiento de los individuos seleccionados para generar nuevos.
5. Reemplazo de los individuos de la población por los recién creados
6. Mutación de individuos para explorar el campo de soluciones

La generación de la población inicial se logra por medio de la función **Genera_Poblacion_Inicial**, esta función recibe como parámetros el tamaño de la población y el tamaño del cromosoma de cada individuo en la población, el valor de cada gen dentro del cromosoma de cada individuo es asignado aleatoriamente (ver Anexo 1, función Genera_Poblacion_Inicial).

La evaluación de cada individuo se calcula con la función **Evaluación**, que recibe como entradas la población, el tamaño de la población, la longitud del cromosoma de cada individuo, un arreglo con el conjunto de palabras que se desea segmentar, un arreglo con todas las subcadenas candidatas a ser prefijos, raíces y sufijos, y el número de posibles prefijos, raíces y sufijos. La evaluación de cada individuo se incrementa según sea menor el número de subcadenas validas que tenga, cada posición del arreglo cuyo valor sea cero y no se tome la subcadena asociada en esa posición para la segmentación, de esta manera se asegura el menor número de subcadenas validas para segmentar las palabras.

Cada concatenación entre combinaciones de subcadenas asociadas a los genes encendidos que formen palabras existentes en la lista de palabras aumentara la evaluación de la posible solución. Además, se toma en cuenta el número de veces que cada subcadena es repetida durante el cálculo de las subcadenas de la lista de palabras, por ejemplo: si una subcadena se repite más veces que la mayoría entre los prefijos, es muy probable que esta subcadena se trate de un prefijo (ver Anexo 1, función Evaluacion). En base a la información mostrada en la figura 4.2 el método puede deducir que la subcadena *a* es un candidato a raíz mas valido que cualquier otro.

El operador genético de selección se ejecuta siguiendo la selección de torneo, esto permite una manera más sencilla de encarar a cada uno de los individuos con los otros, poniéndolos a prueba frente a los demás y asegurando la elección de aquellos con mayor evaluación, la función **Selección** devuelve un arreglo del tipo **Pareja**, donde se guardan los individuos seleccionados por encima de los demás a ser cruzados para la generación de una nueva población (ver Anexo 1, función Seleccion).

Para crear un nuevo conjunto de individuos a partir de los seleccionados mediante el operador genético de selección, se usa el operador genético de cruzamiento, este operador genético se implemento usando un número de bloques y a partir de ese número, se dividen los cromosomas a cruzarse para intercalar su información genética mezclando el número de genes asignados a los bloques que se van a intercalar y de esa manera crear a los nuevos individuos que reemplazaran a aquellos con evaluación menor de la generación actual (ver Anexo 1, función Cruzamiento).

El operador genético de reemplazo, representado por la función **Reemplazo** es encargado de intercambiar a un cierto número de la población actual y reemplazarlos por aquellos individuos recién creados por medio de la función **Cruzamiento**. El reemplazo realizado en esta función **Reemplazo** es elitista, eliminando así a los individuos con menor evaluación y siendo estos reemplazados por aquellos que se acaban de crear.

La función recibe como parámetros la población actual, el nuevo conjunto de individuos apenas creados, el número de individuos que conforman a la población y el porcentaje de reemplazo de individuos a cambiar en la población. Al final, la función retorna la nueva población a ser tomada en cuenta para ser evaluada (ver Anexo 1, función Reemplazo).

Para realizar una búsqueda más amplia a través del espacio de soluciones durante el algoritmo genético, se emplea el operador de mutación, que se encarga de cambiar los genes de un cromosoma al azar, permitiéndole así al algoritmo explorar otras zonas en las que probablemente existan mejores soluciones a las existentes durante la generación en la que se encuentre.

La función encargada de realizar el operador de mutaciones se llama **Mutación** y recibe como parámetros a la población actual del algoritmo, el tamaño de esta población, el tamaño de los cromosomas de los individuos, y el índice de mutación con el que se cuenta durante la generación actual (ver Anexo 1, función Mutacion).

Por último, y para hacer funcionar al algoritmo genético a través de las generaciones se cuenta con la función **Ejecuta_AG** que se encarga de dar valor a sus parámetros y hacer pasar el algoritmo completo a través de los pasos necesarios para estabilizarlo, además de ofrecer la interfaz para mostrar los resultados (ver Anexo 1, función Ejecuta_AG).

4.3 Interfaz de la herramienta

A continuación se muestra la interfaz de la herramienta y la información que se necesita para ejecutar el algoritmo de segmentación.

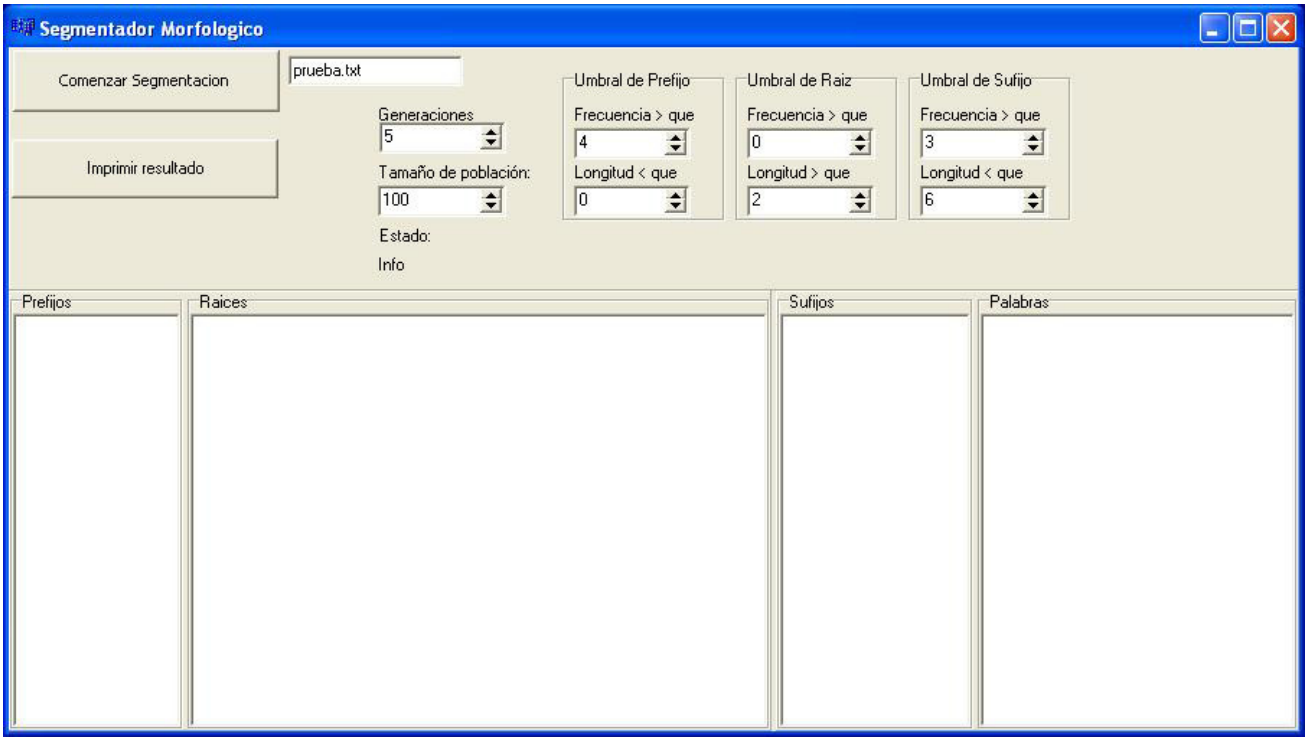


Figura 4.4. Interfaz del sistema de segmentacion

En el cuadro de texto con la leyenda *prueba.txt* se ingresa la ruta del archivo donde se encuentra el conjunto de palabras, en ***Generaciones*** se ingresa el número de generaciones que se ejecutara el algoritmo genético, en ***Tamaño de la población*** se ingresa el número de posibles soluciones sobre las que trabajara el algoritmo.

En los campos de los umbrales de prefijo, raíz y sufijo se indica lo siguiente:

Frecuencia > que

El número de veces mínimo que debe aparecer una subcadena dentro del conjunto al que pertenece para ser tomada en cuenta por el método.

Longitud > que

El tamaño mínimo que debe tener una subcadena para ser tomada en cuenta por el método.

Una vez capturados estos datos de entrada, se da click al botón **Comenzar Segmentacion** para que se inicie el método, después de que este termine,

aparecerán en la lista ***Prefijos*** los prefijos elegidos para la segmentación de las palabras que contengan esa subcadena al inicio de la palabra y de igual forma en las listas de ***Raices*** y ***Sufijos***. En la lista ***Palabras*** aparecerán todas las palabras que existen dentro del archivo que se eligió en el cuadro de texto donde aparece la leyenda *prueba.txt*.

El botón ***Imprimir resultado*** generará un archivo con las palabras segmentadas tal como se muestra en el Anexo 2 de esta tesis.

Capítulo 5. Metodología experimental

El resultado de este trabajo de investigación dio como resultado una herramienta que permite generar todos los posibles segmentos (subcadenas) de un conjunto de palabras en un texto dado y otra (la principal) que calcula los posibles prefijos, raíces y sufijos a través de la frecuencia y validez que tienen éstos con referencia al texto dado. Ambas herramientas están a disposición gratuita para fines académicos.

5.1 Datos utilizados

Las diferentes pruebas que se realizaron fueron hechas con base a la extracción de palabras de textos -como libros electrónicos, tanto en inglés como en español-, diccionarios y palabras aleatorias sin relaciones morfológicas entre ellas, usando entre 15 y 35 palabras como número mínimo de información y 36079 palabras como número máximo de información (pasando entre las 16913, 29088 o 35814 palabras, entre otros).

Los diccionarios utilizados para las pruebas fueron otorgados por el Laboratorio de Lenguaje Natural del Centro de Investigación en Computación del Instituto Politécnico Nacional, contando estos con un promedio de 10000 palabras. Las palabras usadas como pruebas fueron variadas y solo utilizadas para las pruebas iníciales, algunos ejemplos pueden observarse en la tabla 5.1:

Tabla 5.1 Ejemplos de palabras usadas para pruebas iníciales

Abanico
Ábaco
Posiblemente
Invisible
Visiblemente
Probable
Visor
Inmune
Inadvertido
Poblado

Como se puede observar, las palabras para pruebas tienen varias raíces comunes, igual que varios sufijos y varios prefijos en común.

Finalmente, para las pruebas finales, se utilizó el diccionario del sistema AGME [GELBUKH, 02]. En este diccionario, truncamos todas las flexiones, y sólo utilizamos las palabras significativas: sustantivos, verbos y adjetivos. También ignoramos los acentos gráficos, ya que en español su función es ortográfica en la mayoría de los casos.

La lista final tenía 16,849 las palabras sin sus flexiones.

Presentamos a continuación un fragmento de esta lista.

...
alambic
alambicad
alambique
alambrad
alambre
alamed
alarde
alarg
alarid
alarm
alarmant
alarmist
alazan
alb
albañil
albañileri
albace
albahac
albanes
albard
albaricoque
albaricoquer
albatros
albedri
alberg
albergue
albondig
albor
...

5.2 Parámetros del algoritmo

Los experimentos realizados para el sistema de segmentación fueron principalmente basados en los parámetros utilizados para el algoritmo genético y sobre la cantidad de información de la que se disponía en el momento de su ejecución, los parámetros cambiaron hasta llegar a estabilizar el algoritmo.

Para un análisis de los experimentos con los diferentes parámetros se tomaron en cuenta las siguientes medidas relativas al algoritmo en este caso:

Tabla 5.2 Relación de medidas relativas al algoritmo genético para el segmentador de palabras

	Mínimas	**Máximas**
Población	50	1000
Reemplazo	20%	100%
Mutación	20%	90%
Cruzamiento	Punto por punto	20 puntos de cruza
Generaciones	Entre 10 y 50	1000

En su apartado, los resultados se describirán a partir de los valores de los parámetros que se especifican en cada tabla.

Las pruebas iniciales del algoritmo se realizaron con los siguientes valores:

Tabla 5.3 Parámetros para el algoritmo genético en sus primeras pruebas

Población	De 50 a 200
Reemplazo	Del 100%
Mutación	Del 70% al 90%
Puntos de cruzamiento	Punto por punto
Generaciones	entre 100 y 3000

Las siguientes tablas describen los valores utilizados para las pruebas siguientes:

Tabla 5.4 Parámetros para el algoritmo genético en su segunda etapa de pruebas

Población	De 500 a 900
Reemplazo	Del 50%
Mutación	Del 70% al 90%
Puntos de cruzamiento	3
Generaciones	entre 1000 y 3000

Tabla 5.5 Parámetros para el algoritmo genético en su tercera etapa de pruebas

Población	100

Reemplazo	Del 40%
Mutación	20%, disminuyendo a través de las generaciones
Puntos de cruzamiento	4
Generaciones	6000

Tabla 5.6 Parámetros para el algoritmo genético en su cuarta etapa de pruebas

Población	De 100 a 150
Reemplazo	Del 50%
Mutación	Del 70% al 90%
Puntos de cruzamiento	20
Generaciones	entre 1000 y 3000

Tabla 5.7 Parámetros para el algoritmo genético en su quinta etapa de pruebas

Población	200
Reemplazo	Del 80%
Mutación	80%, disminuyendo a través de las generaciones
Puntos de cruzamiento	20
Generaciones	1000

Tabla 5.8 Parámetros para el algoritmo genético en su sexta etapa de pruebas

Población	500
Reemplazo	Del 80%
Mutación	80%, disminuyendo a través de las generaciones
Puntos de cruzamiento	10
Generaciones	7000

Tabla 5.9 Parámetros para el algoritmo genético en su séptima etapa de pruebas

Población	150
Reemplazo	Del 60%
Mutación	30%, disminuyendo a través de las generaciones
Puntos de cruzamiento	5
Generaciones	3000

Con los resultados de los experimentos anteriores se hizo un cambio en el algoritmo; se crea la población inicial y a partir de ésta y con la población que resulta, se ejecuta 3 veces el mismo algoritmo genético, cambiando solamente sus parámetros, se explica cada uno de los 3 pasos:

1. Se regulan las soluciones usando los parámetros de la **tabla 5.9**.

2. Hace una sacudida en la población para ampliar el campo de búsqueda (por el caso en que las soluciones puedan ser mejoradas) usando los parámetros de la **tabla 5.6** cambiando los parámetros del tamaño de la población y aumentando la cantidad de puntos de cruzamiento entre los padres elegidos para incrementar la diversidad genética, quedando los parámetros con los siguientes valores:

3. El algoritmo ahora usa valores en sus parámetros que garanticen la estabilidad de las soluciones en el espacio de búsqueda a través de las generaciones y que gradualmente aumente la evaluación de las soluciones, esto se logra aplicando al algoritmo los valores usados en la **tabla 5.5**.

De esta manera se estabiliza la población inicial, se expande el campo de búsqueda en las soluciones y al final, gradualmente mejoran las soluciones.

5.3 Experimentos

Procesamos los datos presentados más arriba con el algoritmo genético utilizando los parámetros descritos.

No tenemos la manera de evaluar estos resultados de manera automática ya que no existe un estándar para eso para el español. Sin embargo, hicimos una comparación con un sistema que presenta los resultados que corresponden al estado del arte, descritos más abajo.

Adicionalmente, los siguientes resultados fueron obtenidos mientras se preparaban las listas de las cadenas inícials para el algoritmo. En estas, se encontraron 7,747 paradigmas, de los cuales 6,472 contenían más de un elemento. Aquellos paradigmas que contaron con solo un elemento fueron ignorados. Cabe mencionar que de estos 6,472 solo 1,852 paradigmas contenían cadenas vacías. También se utilizó un umbral para la repetición de paradigmas, se ignoraron paradigmas que se repitieron menos de tres veces, es decir, estos paradigmas existen para un máximo de tres palabras o menos. Hubo 5,535 paradigmas con una frecuencia unitaria; 404 con frecuencia igual a dos; y 171 con

frecuencia igual a tres, etc… al final, solo 372 paradigmas fueron elegidos para ser procesados en el algoritmo y estos fueron usados como filtro, esto es, solo los sufijos y raíces que se encuentran en los paradigmas son usados para la representación de los cromosomas. Al final, el algoritmo trabajó con 17,085 raíces y 136 sufijos, los que implicó una reducción importante tomando en cuenta que el número inicial de raíces era mayor a las 44,000 y el mismo número de sufijos fue mayor a los 15,000.

A continuación se muestran los resultados de las divisiones de algunas palabras realizadas con la información resultante del algoritmo para la lista completa. El fragmento más vasto de esta lista se presenta en el Anexo 2.

Las rodeadas por el signo **=** son las raíces detectadas por el algoritmo. La subcadena a la derecha de la raíz es el sufijo de la palabra, de igual manera, en caso de no existir subcadena en esta parte es porque la palabra no posee un sufijo que haya sido detectado en el algoritmo.

Después de los signos **//** van otras posibles variantes de la división. Aunque se debe considerarse la primera como la variante correcta.

Recordamos que las palabras se presentan sin sus flexiones, por ejemplo, la raíz "atac-" corresponde a la palabra "atacar", "atad-" a la palabra "atado", etc.

……
=asunc=ion // =asuncion=
=asu=nt
=asu=st // =asust=
=at=
=atañ=
=atac=
=atac=ant // =ataca=nt
=atac=ante // =ataca=nte // =atacante=
=atad=
=atadu=r
=ataj=
=atalay=
=ataq=ue // =ataqu=e // =ataque=
=atardec=
=atardece=r
=ataread=
=atas=c // =atasc=
=atasca=der // =atascad=er // =atascade=r // =atascader=
…

Los resultados muestran una concordancia entre las divisiones de palabras y la información sobre la que se basó el algoritmo para su segmentación de cada una de ellas.

5.4 Evaluación de los resultados

Se compararon estos resultados con los resultados que arroja el sistema *Lingüística*, dando como entrada los mismos datos que utilizamos. Desafortunadamente, no contamos con el mismo estándar para la evaluación automática de nuestros datos. La evaluación manual de los resultados muestra que este sistema produce una división comparable de palabras durante el procedimiento de evaluación que a continuación se describe.

La versión de *Linguistica* que tenemos acepta solamente 5,000 palabras como datos de entrada. Se hizo otro experimento y se dieron los mismos datos de entrada en ambos sistemas: las primeras 5,000 palabras de nuestro diccionario. Este número de palabras debe ser suficiente en ambos sistemas porque en ambos se usa el aprendizaje no supervisado. Después de evaluarse manualmente las divisiones obtenidas en las primeras cien palabras se obtuvo el siguiente resultado: *Linguistica* tuvo un 87% de precisión mientras que el nuestro sistema produce un 84%.

Tabla 5.10 Evaluación preliminar de la precisión de los sistemas

	Linguistica	Sistema propuesto
Precisión	87%	84%

Recordemos que el sistema *Linguistica* hace unas suposiciones sobre los datos que este método no hace, ganando así una independencia para la adquisición de conocimiento acerca del problema mientras trabaja.

Ambos sistemas obtuvieron errores en diferentes palabras, por ejemplo, *Linguistica* no encontró el sufijo *–mient(o)* que fue frecuente en la lista y fue detectado por nuestro sistema. Obviamente estos valores corresponden solo a una estimación preliminar. Estos valores pueden parecer muy altos debido a que el estándar de los sistemas actuales ronda cerca del 60%, pero cabe recordar que se está tratando con morfología derivativa y que estamos trabajando con un diccionario, no con un corpus.

Capítulo 6. Conclusiones

6.1 Conclusiones

En este trabajo se presenta un algoritmo no supervisado capaz de evaluar las diferentes estructuras en la que se forman las palabras con respecto al entorno en la que se encuentran, con esta información es posible determinar las unidades mínimas capaces de concatenarse y mantenerse válidas dentro del contexto al que pertenecen, un efecto similar es el que ocurre en el mundo real al hacer referencias a morfos y palabras existentes en un lenguaje. Utilizamos el concepto del paradigma para reducir las posibles soluciones del algoritmo.

El algoritmo es capaz de realizar segmentaciones exitosas dependiendo de la cantidad de información disponible, en los casos en los que se le entregue un conjunto de palabras reducido y sin conexiones morfológicas entre ellas, los resultados se verán afectados en segmentación muy poco exitosas.

La evaluación preliminar del algoritmo muestra que tiene la precisión comparable con los métodos descritos en el capitulo 2.2.4. Sin embargo, nuestro método no hace ninguna suposición a priori presente en los métodos del estado del arte, por ejemplo, se supone la distribución de probabilidades de Boltzman, lo que es una muy buena heurística ya que se reduce el espacio de soluciones, pero eso significa que la formulación del problema ya no es la misma. Entonces, nuestro método es mucho más orientado a la realidad lingüística como tal, sin suposiciones adicionales.

6.2 Aportaciones

Se desarrolló una herramienta que divide las palabras en tres segmentos; raíz, prefijo y sufijo, basada en el trabajo realizado anteriormente por Alexander Gelbukh, Mikhail Alexandrov y Sang Yong Han en [GELBUKH, 04] en el que su metodología, como se mencionó en el capítulo 2 de esta tesis, detecta los conjuntos mínimos de posibles raíces y terminaciones en un conjunto de palabras.

La metodología usada para la realización de este trabajo estuvo fuertemente basada en la de [GELBUKH, 04]. Nosotros adicionamos este enfoque con la idea de aplicar el concepto de los paradigmas, ya que sin este concepto el método da muy malos resultados aplicado a la morfología derivativa. En el artículo original, el método se aplica a la las flexiones únicamente, y no a los afijos. Cabe mencionar que los afijos tienen menor frecuencia que las flexiones. De esta manera, aplicamos el método a nivel derivativo, es decir, tratando de buscar afijos, y no las flexiones.

6.3 Trabajo futuro

6.3.1 Uso de la herramienta en diferentes lenguajes

Por ser el español un lenguaje relativamente simple para la formación y creación de palabras, las diferentes unidades morfológicas son fácilmente reusables para estas tareas, sin embargo existen lenguajes en los que se combinan morfemas para la creación de palabras muy grandes, como el inuktitut que utiliza el aglutinamiento de morfemas para transmitir ideas enteras, en algunas ocasiones, en unidades de una sola palabra, por ejemplo:

qasuiirsarvigssarsingitluinarnarpuq

qasu	cansado
iir	no
sar	causa
vig	lugar para
ssar	apropiado
si	encontrar
ngit	no
luinar	completamente
nar	alguien
puq	tercera persona

"Alguien no encontró un lugar de descanso completamente apropiado."

El finlandés es también un buen ejemplo para mostrar este tipo de aglutinamiento morfológico.

El turco es también uno de los lenguajes que define claramente cada uno de sus morfemas dentro la palabra, por lo que su detección es sencilla.

Se podría modificar la metodología desarrollada en este trabajo para adaptarla para este tipo de lenguajes y localizar cada uno de los morfemas que conforman sus palabras. Una estructura recursiva sobre los resultados que arroja la metodología desarrollada en esta tesis podría ser una solución a este problema.

Se hicieron algunas pruebas sobre escritos en inglés, y debido a las formas gramaticales que este adopta sobre las flexiones en sus palabras, tan parecidas al español, los resultados fueron menos favorables que los mostrados en esta tesis, además de que, como en el español, se ignoraron ciertas reglas para el tratamiento de las flexiones irregulares, como los cambios singular – plural de algunas palabras como: Wolf – Wolves, Knife – Knives ó Sky – Skies. Parte del trabajo por hacer es incluir las reglas que trabajan sobre cambios en los valores gramaticales de las palabras.

6.3.2 Separación de palabras en mayor número de morfemas

Como ya se menciono, las separaciones de palabras en este trabajo se dan en tres partes; prefijo, raíz y sufijo, sin embargo, al separar una palabra en morfemas existen casos en los que las partes de las palabras son más que estas debido a que cada morfo en una palabra representa una o varias ideas gramaticales, tomemos por ejemplo la palabra **trabajadores**, su división morfológica sería la siguiente:

trabaj-a-dor-es

Donde la raíz de la palabra es **tabaj**, mientras que sus demás unidades morfológicas son *a*, *dor* y *es*, las cuales pueden ser sustituidas por otras alternado su significado, por lo que podrían resultar las siguientes palabras:

trabaj-a-Ø-Ø
trabaj-a-dor-Ø
trabaj-o-Ø-Ø
trabaj-o-s-Ø
trabaj-a-dor-as
trabaj-é-Ø-Ø

Por solo mencionar unos ejemplos.

Para el desarrollo de una metodología capaz de hacer lo anteriormente descrito es necesario un análisis más profundo sobre el lenguaje sobre el que se

desee aplicar. Un siguiente paso sobre este trabajo es el poder dividir las palabras en morfos lingüísticos más completos, como los descritos con este ejemplo.

Bibliografía

[ALPHADICTIONARY. 06] http://www.alphadictionary.com/articles/ling005.html, "How many words are in English? – alphaDictionary", 14-01-06

[BARONI et al., 02] Baroni, M., Matiasek, J., Trost, H.: Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In: ACL Workshop on Morphological and Phonological Learning (2002)

[BOLSHAKOV, 00] BOLSHAKOV, I. y GELBUKH, A. Computational Linguistics and Linguistic Models. CIC-IPN. ISBN 970-36-0147-2, México, 2004.

[BOSQUE, 99] BOSQUE, I. DEMONTE, V. Gramática Descriptiva de la Lengua Española (tomo 3). Entre la Oración y el Discurso - Morfología. Editorial Espasa. España, 1999.

[COLE, 96] COLE, Ronald. et. al. Survey of the State of the Art in Human Language Technology. EEUU., Cambridge University Press, 1996. 530 p.

[CREUTZ, 03] Creutz, Mathias: Unsupervised Segmentation of Words Using Prior Distributions of Morph Length and Frequency. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03), Sapporo, Japan, pp 280-287 (2003).

[CRYSTAL, 91] CRISTAL, D. A Dictionary of Linguistics and Phonetics. 3a. Ed. Ed. Blackwell. EEUU., 1991.

[GALICIA, 00] GALICIA Haro, Sofía N. Análisis Sintáctico conducido por un Diccionario de Patrones de Manejo Sintáctico para Lenguaje Español. Tesis (Doctorado en Ciencias de la Computación). México, D.F., México, Instituto Politécnico Nacional, Centro de Investigación en Computación, 2000. 335 p.

[GELBUKH, 02] GELBUKH, A. y SIDOROV, G. Morphological Analysis of

Inflective Languages trough Generation. Centro de Investigación en Computación, Instituto Politécnico Nacional, México, D. F., 2002.

[GELBUKH, 04] GELBUKH, A., ALEXANDROV, M. y SANG YONG HAN. *Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model*. Centro de Investigación en Computación, Instituto Politécnico Nacional, México, D. F. y Chung-Ang University Korea, 2004.

[GOLDSMITH, 91] Goldsmith, J. Unsupervised Learning of the Morphology of a Natural Language. Computational Linguistics 27:2, 153-198 (2001)

[GRICE, 89] GRICE, P. Studies in the Way of Words. Ed. Harvard Univ. EE.UU., 1989.

[Haahr, 07] Haahr P., Baker S.: Making search better in Catalonia, Estonia, and everywhere else. Google blog, http://googleblog.blogspot.com/2008/03/making-search-better-in-catalonia.htm (2007)

[KAZAKOV, 97] Kazakov D.: Unsupervised learning of naıve morphology with genetic algorithms. // Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks. Prague, Czech Republic, pp. 105–112 (1997)

[MORPHOCHALLENGE, 05] Pascal Morphochallenge-2005, http://www.cis.hut.fi/morphochallenge2005/

[MORPHOCHALLENGE, 07] Pascal Morphochallenge-2007, http://www.cis.hut.fi/morphochallenge2007/

[NIDA, 86] NIDA, E. Sociolinguistics and Translating. Ed. Sociolinguistics and communication. Londres, 1986.

[REHMAN, 00] Rehman, Kh., Hussain, I.: Unsupervised Morphemes Segmentation. In: Pascal Morphochallenge, 5 p. (2005)

ANEXO 1. Codigo fuente

En este anexo se muestra el codigo fuente escrito en el lenguaje de programacion Borland C++ Builder 6 para efecto de comprension de quien lo necesite.

Funcion **Segmenta_Palabras**:

```
void Segmenta_Palabras (String * strPalabras,
                        int     intNumero_Palabras,
                        String * & Lista_de_Cadenas,  // Eso se asigna adentro, los 5 siguientes
                        int    * & Lista_de_Frec,
                        int    & intPrefijos,
                        int    & intRaices,
                        int    & intSufijos,
                        int    Frec_Umbral_Prefijo,
                        int    Frec_Umbral_Raiz,
                        int    Frec_Umbral_Sufijo,
                        int    Leng_Umbral_Prefijo,
                        int    Leng_Umbral_Raiz,
                        int    Leng_Umbral_Sufijo,
                       THash_map_type * & HMPrefijos_Umbral,
                       THash_map_type * & HMRaices_Umbral,
                       THash_map_type * & HMSufijos_Umbral
                       )
{
   String strPalabra;

   String strPrefijo;
   String strRaiz;
   String strSufijo;

                       TFrecYPos FrecYPos;
                       FrecYPos.frec = 1;
                       FrecYPos.pos  = 0;

                       THash_map_type * HMPrefijos = new THash_map_type ();    // Here we insert all
possible data
   THash_map_type * HMRaices   = new THash_map_type ();
   THash_map_type * HMSufijos  = new THash_map_type ();

   //ciclo que ingresa palabra por palabra todas las subcadenas posibles
   for (int intPalabra = 0; intPalabra < intNumero_Palabras; intPalabra++ )
   {
      strPalabra = strPalabras [intPalabra];

        HMPrefijos->insert (THash_map_type::value_type ("", FrecYPos));  // Always present
        HMSufijos ->insert (THash_map_type::value_type ("", FrecYPos));
```

```
//ya con la palabra "limpia" se agregan todos sus prefijos, raices y sufijos
        for (unsigned i = 1; i <= strPalabra.Length () - 1; i++)  // -1 = Raiz no vacio,
        {
                strPrefijo = strPalabra.SubString (1, i).c_str ();
                HMPrefijos->insert (THash_map_type::value_type (strPrefijo.c_str(), FrecYPos));
        }

        for (unsigned j = 1; j <= strPalabra.Length (); j++)
        {
          strRaiz    = strPalabra.SubString (1, j).c_str ();             // 2 veces, de principio y de fin
         HMRaices  ->insert (THash_map_type::value_type (strRaiz.c_str(), FrecYPos));

         if (Leng_Umbral_Prefijo > 1)
         {
                  strRaiz    = strPalabra.SubString (j, strPalabra.Length()).c_str ();
                  HMRaices  ->insert (THash_map_type::value_type (strRaiz.c_str(), FrecYPos));
         }
         }

        for (unsigned k = 1 + 1; k <= strPalabra.Length (); k++)
        {
                strSufijo  = strPalabra.SubString (k, strPalabra.Length ()).c_str ();
                HMSufijos ->insert (THash_map_type::value_type (strSufijo.c_str(), FrecYPos));
        }
}//fin for que avanza por el arreglo "Palabras"

    int intPrefijo;
    int intRaiz;
    int intSufijo;

   pair <THash_map_type::const_iterator, THash_map_type::const_iterator> pairPrefijo;
    pair <THash_map_type::const_iterator, THash_map_type::const_iterator> pairRaiz;
    pair <THash_map_type::const_iterator, THash_map_type::const_iterator> pairSufijo;

    // ciclo que ingresa palabra por palabra todas las subcadenas posibles,
    // CALCULA FRECUENCIAS y Filtra por Length y Frec
    for (int intPalabra = 0; intPalabra < intNumero_Palabras; intPalabra++ )
    {
       strPalabra = strPalabras [intPalabra];

       //ya con la palabra "limpia" se agregan todos sus prefijos, raices y sufijos
       for (unsigned i = 0; i <= strPalabra.Length () - 1; i++)  // -1 = Raiz no vacio, Empieza 0 = se
necesita cadena vacia
       {
          strPrefijo = strPalabra.SubString (1, i).c_str ();

          if (i > 0 && Leng_Umbral_Prefijo <= 1)   // Solo se considera' prefijo vacio
             continue;

          for (unsigned j = i + 1; j <= strPalabra.Length (); j++)
          {
             strRaiz    = strPalabra.SubString (i + 1, j - i).c_str ();                    // La raiz
```

```
    strSufijo  = strPalabra.SubString (j + 1, strPalabra.Length ()).c_str ();
                // El resto de la palabra es sufijo

    if (strPrefijo != ""
    &&  (   (strPrefijo[strPrefijo.Length()] == 'l' && strRaiz [1] == 'l')  // No dividir "ll" y "ch"
        || (strPrefijo[strPrefijo.Length()] == 'c' && strRaiz [1] == 'h') )
      )
      continue;

    if (strSufijo != ""
    &&  (   (strRaiz[strRaiz.Length()] == 'l' && strSufijo [1] == 'l')
        || (strRaiz[strRaiz.Length()] == 'c' && strSufijo [1] == 'h') )
      )
      continue;

    bool Existe_Prefijo = false;
    bool Existe_Raiz    = false;
    bool Existe_Sufijo  = false;

    //se cuentan las veces que se encontro cada subcadena en el total
    intPrefijo = HMPrefijos->count (strPrefijo.c_str());
    intRaiz    = HMRaices  ->count (strRaiz.c_str());
                 intSufijo  = HMSufijos ->count (strSufijo.c_str());

//se calculan los rangos (en caso de encontrarse) en los que
    //esta ubicada cada subcadena en el filtro del umbral
    pairPrefijo = HMPrefijos_Umbral->equal_range (strPrefijo.c_str());
    pairRaiz    = HMRaices_Umbral  ->equal_range (strRaiz.c_str());
    pairSufijo  = HMSufijos_Umbral ->equal_range (strSufijo.c_str());

    //se busca cada subcadena para no repetirla en el filtro del umbral
    THash_map_type::const_iterator THMIterador;

    for (THMIterador = pairPrefijo.first; THMIterador != pairPrefijo.second; THMIterador++)
       Existe_Prefijo = true;

for (THMIterador = pairRaiz.first; THMIterador != pairRaiz.second; THMIterador++)
       Existe_Raiz = true;

    for (THMIterador = pairSufijo.first; THMIterador != pairSufijo.second; THMIterador++)
       Existe_Sufijo = true;

    //en caso de no encontrarse, que cada subcadena sobrepase el umbral, y
    //que los prefijos y sufijos no sean muy largos...
    //entonces se agrega para ser tomada en cuenta
   if (!Existe_Prefijo
   &&  (intPrefijo > Frec_Umbral_Prefijo || strPrefijo == "") // Siempre guardamos prefijo ""
    &&  (strPrefijo.Length () < Leng_Umbral_Prefijo || strPrefijo == "")  )
{
    FrecYPos.frec = intPrefijo;
   HMPrefijos_Umbral->insert (THash_map_type::value_type (strPrefijo.c_str(), FrecYPos));
    intPrefijos++;
  }
```

```
          if (!Existe_Raiz
          &&  intRaiz > Frec_Umbral_Raiz
          &&  strRaiz.Length() > Leng_Umbral_Raiz)  // En caso de raiz es >, y no menor
          {
           FrecYPos.frec = intRaiz;
          HMRaices_Umbral->insert (THash_map_type::value_type (strRaiz.c_str(), FrecYPos));
          intRaices++;
          }

          if (!Existe_Sufijo
          &&  (intSufijo > Frec_Umbral_Sufijo  || strSufijo == "")  // Siempre guardamos sufijo ""
          &&  (strSufijo.Length() < Leng_Umbral_Sufijo || strSufijo == "")  )
      {
        FrecYPos.frec = intSufijo;
        HMSufijos_Umbral->insert (THash_map_type::value_type (strSufijo.c_str(), FrecYPos));
          intSufijos++;
       }
      }
    }
}//fin for que avanza por el arreglo "Palabras"

HMPrefijos->clear ();  // Todos datos en .._Umbral
HMRaices  ->clear ();
HMSufijos ->clear ();

delete HMPrefijos;
delete HMRaices;
delete HMSufijos;

//almacenador de todos los posibles prefijos, raices y sufijos
Lista_de_Cadenas = new String [(intPrefijos + intRaices + intSufijos)];
Lista_de_Frec    = new int    [(intPrefijos + intRaices + intSufijos)];

//indicara la posicion en la que se asignara el prefijo, raiz o sufijo en el
//arreglo "Lista_de_Cadenas"
int intPosicion = 0;
      THash_map_type::iterator i;

      //se le ingresan todos los posibles prefijos
      for (i = HMPrefijos_Umbral->begin (); i != HMPrefijos_Umbral->end (); i++)
      {
               Lista_de_Cadenas [intPosicion] = (*i).first.c_str ();
               Lista_de_Frec    [intPosicion] = (*i).second.frec;
               (*i).second.pos = intPosicion;
               intPosicion++;
      }

//se le ingresan todos las posibles raices
 for (i = HMRaices_Umbral->begin (); i != HMRaices_Umbral->end(); i++)
{
   Lista_de_Cadenas [intPosicion] = (*i).first.c_str ();
   Lista_de_Frec    [intPosicion] = (*i).second.frec;
```

```
      (*i).second.pos = intPosicion;
      intPosicion++;
   }

   //se le ingresan todos los posibles sufijos
   for (i = HMSufijos_Umbral->begin (); i != HMSufijos_Umbral->end(); i++)
   {
        Lista_de_Cadenas [intPosicion] = (*i).first.c_str ();
        Lista_de_Frec    [intPosicion] = (*i).second.frec;
        (*i).second.pos = intPosicion;
        intPosicion++;
   }
}
```

Después de segmantar palabras, estamos obteniendo los **paradigmas** y filtramos los elementos que no forman paradigmas.

```
TFrecYPos FrecYPos;  // En Palabras frecuencias no se usan
FrecYPos.frec = 1;
FrecYPos.pos  = 0;

for (int intPalabra = 0; intPalabra < num_Palabras; intPalabra++)
    HMPalabras->insert (THash_map_type::value_type (strPalabras [intPalabra].c_str (), FrecYPos));

for (int i = Prefijos;  i < Prefijos + Raices; i++) // Ciclo para todos los raices, formamos paradigmas
   {
     String raiz = Subcadenas [i];
     String parad = "";

     for (int j = Prefijos + Raices;  j < Prefijos + Raices + Sufijos; j++)  // Ciclo para sufijos
     {
       String sufijo = Subcadenas [j];

       String palabra = raiz + sufijo;

       pair <THash_map_type::const_iterator, THash_map_type::const_iterator> pairPal;

       pairPal = HMPalabras->equal_range (palabra.c_str ());

       if (pairPal.first != pairPal.second) // Existe palabra
       {
         if (sufijo == "")
           sufijo = "NULL";

         parad = parad + "," + sufijo;
       }
     }

     parad.Delete (1, 1);  // Eliminar coma al inicio

     pair <THash_map_type::iterator, THash_map_type::iterator> pairParad;
```

```
  pairParad = HMParadigmas->equal_range (parad.c_str ());

  if (pairParad.first != pairParad.second) // Existe paradigma
  {
     THash_map_type::iterator HMIterador;
     HMIterador = pairParad.first;

     (*HMIterador).second.frec++;
  }
  else
  {
    FrecYPos.frec = 1;
    HMParadigmas->insert(THash_map_type::value_type (parad.c_str (), FrecYPos));
  }
}

ofstream f;
f.open ("paradigm.txt");   // Imprimir los datos en el archivo

for (THash_map_type::iterator i =  HMParadigmas->begin();
                                i != HMParadigmas->end(); i++)
{
  String parad = (*i).first.c_str();
  int pos  = (*i).second.pos;
  int frec = (*i).second.frec;

  f << frec << " " << pos << " " << parad.c_str() << endl;
}

f.close ();

// Limpiar paradigmas
for (THash_map_type::iterator it =  HMParadigmas->begin(); it != HMParadigmas->end(); it++)
{
   String parad = (*it).first.c_str();

   String Primero;
   if (parad.Pos(",") > 0)
   {
     Primero= parad.SubString (1, parad.Pos(",") - 1);
     parad.Delete (1, parad.Pos(","));
   }
   else
   {
      Primero = parad;
      parad = "";
   }

   String Segundo = "";

   if (parad.Pos(",") > 0)
   {
```

```
            Segundo = parad.SubString (1, parad.Pos(",") - 1);
            parad.Delete (1, parad.Pos(","));
        }
        else
        {
            Segundo = parad;
            parad =  "";
        }

        if ( (Segundo == "" && Primero != "NULL")    // Un solo elemento, sin contar NULL
//          ||   (      parad == ""  && Primero != "NULL" && Segundo != "NULL"  //    Paradigma de 2,
uno es una letra ; Mas abajo tambien verificamos UNA letra
//              && (Primero.Length() == 1 || Segundo.Length () == 1) )
            )
            (*it).second.frec = 0;
    }

    // ****************** Eliminar elementos que no forma paradigmas
     String * subcad_temp = new String [Sufijos + 1];

     subcad_temp [0] = "";              // Agregamos NULL siempre
     int new_suf = 1;

     for (int i = Prefijos + Raices; i < Prefijos + Raices + Sufijos; i++)
     {
       String s = Subcadenas [i];

       bool Present = false;

     for (THash_map_type::iterator it =  HMParadigmas->begin(); it != HMParadigmas->end(); it++)
     {
          if ((*it).second.frec <= Umbral_frec_paradigmas)
            continue;

          String parad = (*it).first.c_str();

          if (parad.Pos(s + ",") == 1      // Primera posicion
          ||  parad == s
          ||  parad.Pos("," + s + ",") > 0  // en medio
          ||  (   parad.Length () - parad.Pos("," + s) == s.Length ())
               && parad.Pos("," + s) > 0) // ultima
          {
            Present = true;
            break;
          }
       }

       if (Present)
       {
         subcad_temp    [new_suf] = s;
         new_suf++;
       }
     }
```

```
    Sufijos = new_suf;

    for (int i = 0; i < Sufijos; i++)
        Subcadenas [Prefijos + Raices + i] = subcad_temp [i];

    delete [] subcad_temp;

    THash_map_type * HMSufijos_new = new THash_map_type();

    for (int i = Prefijos + Raices; i < Prefijos + Raices + Sufijos; i++)
    {
      String s = Subcadenas [i];

      pair <THash_map_type::const_iterator, THash_map_type::const_iterator> pairSufijo;
      pairSufijo  = HMSufijos_Umbral->equal_range (s.c_str());

      for (THash_map_type::const_iterator HMIterador = pairSufijo.first; HMIterador != pairSufijo.second; HMIterador++)
      {
        Subcadenas_frec [i] = (*HMIterador).second.frec;

        FrecYPos.frec = 0;   //Se asignara despues
        FrecYPos.pos  = i;

        HMSufijos_new->insert(THash_map_type::value_type (s.c_str (), FrecYPos));
      }
    }

    HMSufijos_Umbral->clear ();

    for (THash_map_type::iterator it =  HMSufijos_new->begin(); it != HMSufijos_new->end(); it++)
      HMSufijos_Umbral->insert(THash_map_type::value_type ((*it).first, (*it).second));

    HMSufijos_new->clear ();
    delete HMSufijos_new;

  // ****************** Eliminar RAICES que no forma palabras con nuevos sufijos

  THash_map_type * HMRaices_new = new THash_map_type();


    for (THash_map_type::iterator it_raiz =  HMRaices_Umbral->begin();
          it_raiz != HMRaices_Umbral->end();
          it_raiz++)
  {

       String raiz = (*it_raiz).first.c_str();


       for (THash_map_type::iterator it_suf =  HMSufijos_Umbral->begin();
            it_suf != HMSufijos_Umbral->end();
```

```
            it_suf++)
        {

         String sufijo =  (*it_suf).first.c_str();

         String palabra = raiz + sufijo;

         pair <THash_map_type::iterator, THash_map_type::iterator> pairPalabra;

         pairPalabra  = HMPalabras->equal_range (palabra.c_str());

         if (pairPalabra.first != pairPalabra.second)        // Exista la palabra
         {
           // Cambiar frec de raices y sufijos
           pair <THash_map_type::iterator, THash_map_type::iterator> pairRaiz_new;
           pairRaiz_new  = HMRaices_new->equal_range (raiz.c_str());

           if (pairRaiz_new.first != pairRaiz_new.second)        // Exista la raiz
           {
              (*pairRaiz_new.first).second.frec++;
           }
           else
           {
             FrecYPos.pos  = (*it_raiz).second.pos;
             FrecYPos.frec = 1;
             HMRaices_new->insert(THash_map_type::value_type (raiz.c_str (), FrecYPos));
           }

           pair <THash_map_type::iterator, THash_map_type::iterator> pairSufijo;
           pairSufijo  = HMSufijos_Umbral->equal_range (sufijo.c_str());

           if (pairSufijo.first != pairSufijo.second)        // Exista el sufijo
           {
              (*pairSufijo.first).second.frec++;
           }
         }
        }
    }


    HMRaices_Umbral->clear ();
    int cnt = 0;

    for (THash_map_type::iterator it =   HMRaices_new->begin(); it != HMRaices_new->end();
it++)
    {
      (*it).second.pos = Prefijos + cnt;
      HMRaices_Umbral->insert(THash_map_type::value_type ((*it).first, (*it).second));

      Subcadenas      [Prefijos + cnt] = (*it).first.c_str();
      Subcadenas_frec [Prefijos + cnt] = (*it).second.frec;

      cnt++;
```

```
    }

    Raices = cnt;

    HMRaices_new->clear ();
    delete HMRaices_new;

    cnt = Prefijos + Raices;        // Ahora mover sufijos en Subcadenas, ya que son menos Raices

    for (THash_map_type::iterator it =  HMSufijos_Umbral->begin();
         it != HMSufijos_Umbral->end();
         it++)
    {
      (*it).second.pos = cnt;

      Subcadenas      [cnt] = (*it).first.c_str();
      Subcadenas_frec [cnt] = (*it).second.frec;

      cnt++;
    }
```

Estructura **Cromosoma**:

```
struct Cromosoma
{
   int   * Gen;
   float   Evaluacion;
   int     Probabilidad_Mutacion;
   int     Probabilidad_Reemplazo;
};
```

Estructura **Pareja**:

```
struct Pareja
{
   int Padre;
   int Madre;
};
```

Estructura **TFrecYPos**:

```
struct TFrecYPos
{
         int frec;
         int pos;
};
```

Funcion **Dimensiona_Poblacion**:

```
Cromosoma * Dimensiona_Poblacion (int Tamano_Poblacion,
                                  int Tamano_Cromosoma)
{
        Cromosoma * Individuos_Poblacion;

        //hace el arreglo tan grande como individuos sean en la poblacion
        Individuos_Poblacion = new Cromosoma [Tamano_Poblacion];

        for (int Individuo = 0; Individuo < Tamano_Poblacion; Individuo++)
        {
                //a cada individuo le asigna su tamaño de cromosoma
                Individuos_Poblacion [Individuo].Gen = new int [Tamano_Cromosoma];
        }

        return Individuos_Poblacion;
}
```

Funcion **Torneo**:

```
int Torneo (Cromosoma * Individuos,
            int         Tamano_Poblacion)
{
        int Aleatorio1;
        int Aleatorio2;

        //se eligen 2 individuos al azar
        Aleatorio1 = (rand () % Tamano_Poblacion);
        Aleatorio2 = (rand () % Tamano_Poblacion);

        //para que los padres a elegir no sea el mismo individuo
        while (Aleatorio1 == Aleatorio2)
        {
                Aleatorio2 = (rand () % Tamano_Poblacion);
        }

        //se realiza el torneo entre ambos individuos para elegir el mejor
        if (Individuos [Aleatorio1].Evaluacion > Individuos [Aleatorio2].Evaluacion)
        {
                return Aleatorio1;
        }
        else
        {
                return Aleatorio2;
        }
}
```

Funcion **Ordena_Poblacion**:

```
Cromosoma * Ordena_Poblacion (Cromosoma * Individuos,
                              int          Tamano_Poblacion)
{
        Cromosoma Individuo_Aux;

        //La poblacion se ordena con el metodo de la burbuja
        //la ordenacion es de forma ascendiente, el individuo
        //con mayor evaluacion queda al final
        for (int Principio = 1; Principio < Tamano_Poblacion; Principio++)
        {
                for (int Final = (Tamano_Poblacion - 1); Final > Principio; Final--)
                {
                        if (Individuos [Final - 1].Evaluacion > Individuos [Final].Evaluacion)
                        {
                                Individuo_Aux          = Individuos [Final - 1];
                                Individuos [Final - 1] = Individuos [Final];
                                Individuos [Final]     = Individuo_Aux;
                        }
                }
        }

        return Individuos;
}
```

Funcion **Genera_Poblacion**:

```
Cromosoma * Genera_Poblacion_Inicial (int Tamano_Poblacion,
                                      int Tamano_Cromosoma)
{
        Cromosoma * Individuos_Poblacion_Inicial;

        //se crean los individuos en "blanco"
        Individuos_Poblacion_Inicial = Dimensiona_Poblacion (Tamano_Poblacion,
                                                             Tamano_Cromosoma);

        randomize ();

        //se asigna los valores de los genes aleatoriamente
        for (int Individuo = 0; Individuo < Tamano_Poblacion; Individuo++)
        {
                for (int Gen = 0; Gen < Tamano_Cromosoma; Gen++)
                {
                        //si el numero aleatorio es par, al gen se le asigna un 1
                        if (((rand () % 100) % 2) == 0)
                                Individuos_Poblacion_Inicial [Individuo].Gen [Gen] = 1;
                        //si el numero aleatorio es impar, al gen se le asigna un 0
                        else
                                Individuos_Poblacion_Inicial [Individuo].Gen [Gen] = 0;
                }//fin del For que recorre cada Gen
        }//fin de For que recorre a cada individuo

        return Individuos_Poblacion_Inicial;
```

```
}
```

Funcion **Evaluacion**:

```
void Evaluacion (Cromosoma      * Individuos,
                 int            intTamano_Poblacion,
                 int            intTamano_Cromosoma,   // Pref + Raices + Suf
                 THash_map_type * HMPalabras,
                 String         * Subcadenas,
                 int            intPrefijos,
                 int            intRaices,
                 int            intSufijos,
                  THash_map_type * HMPrefijos,
                  THash_map_type * HMRaices,
                  THash_map_type * HMSufijos,
                  TLabel         * Info,
                  TApplication   * App
                 )
{
   for (int intIndividuo = 0; intIndividuo < intTamano_Poblacion; intIndividuo++)
   {
      int cnt = 1;

      for (int intGen = 0; intGen < intTamano_Cromosoma; intGen++)
      {
         //entre menos prefijos, raices y sufijos tenga un individuo, su
         //fitness es mayor
         if (Individuos [intIndividuo].Gen [intGen] == 0)
            cnt++;
      }//fin for de genes

      Individuos [intIndividuo].Evaluacion =  -log (1.0 / cnt) / intTamano_Cromosoma;
   }//fin for de individuos


  for (int intIndividuo = 0; intIndividuo < intTamano_Poblacion; intIndividuo++)
  {
     THash_map_type::const_iterator p;
      String strSufijo, strPrefijo, strRaiz;

      for (p = HMPalabras->begin (); p != HMPalabras->end (); p++)
      {
          String strPalabra  = (*p).first.c_str ();

      for (unsigned i = 0; i <= strPalabra.Length () - 1; i++)  // -1 = Raiz no vacio, i = 0 prefijo vacio
      {
           strPrefijo = strPalabra.SubString (1, i);

           for (unsigned j = i + 1; j <= strPalabra.Length (); j++)
           {
              strRaiz    = strPalabra.SubString (i + 1, j - i);
              strSufijo  = strPalabra.SubString (j + 1, strPalabra.Length ());
```

```
          // si la palabra existe, el fitness del individuo aumenta
          pair <THash_map_type::const_iterator, THash_map_type::const_iterator>

         pairExiste_pref, pairExiste_raiz, pairExiste_suf;

          pairExiste_pref = HMPrefijos->equal_range (strPrefijo.c_str ());

          if (pairExiste_pref.first != pairExiste_pref.second)
          {
             pairExiste_suf = HMSufijos->equal_range (strSufijo.c_str ());

            if (pairExiste_suf.first != pairExiste_suf.second)
            {
               pairExiste_raiz = HMRaices->equal_range (strRaiz.c_str ());

               if (pairExiste_raiz.first != pairExiste_raiz.second)
               {
                  THash_map_type::const_iterator iter = pairExiste_pref.first;
                   int posPref  = (*iter).second.pos;  // Es la posicion de elemento en cada cromosoma
                   int frecPref = (*iter).second.frec;

                    iter = pairExiste_raiz.first;
                    int posRaiz  = (*iter).second.pos;
                    int frecRaiz = (*iter).second.frec;

                    iter = pairExiste_suf.first;
                     int posSuf  = (*iter).second.pos;
                     int frecSuf = (*iter).second.frec;

                     if (Individuos [intIndividuo].Gen   [posPref] == 1
                     &&       Individuos [intIndividuo].Gen   [posRaiz] == 1
                     &&       Individuos [intIndividuo].Gen   [posSuf]  == 1)
                     {

                             Individuos [intIndividuo].Evaluacion
                                   += (-log (1.0 / frecPref)
                                        -log (1.0 / frecRaiz)
                                        -log (1.0 / frecSuf)) / intTamano_Cromosoma;
                      }
                     }
                 }
               }
            }
         }
      }

      Info->Caption = "Individuo " + IntToStr (intIndividuo);
      App->ProcessMessages ();
                  }//fin for de Individuos
}
```

Funcion **Seleccion**:

```
Pareja * Seleccion (Cromosoma * Individuos,
                    int          Tamano_Poblacion,
                    int          Porcentaje_Reemplazo)
{
        Pareja * Padres;
        int      Reemplazo;

        Reemplazo  = (Porcentaje_Reemplazo * Tamano_Poblacion) / 100;
        Padres     = new Pareja [Reemplazo];

        for (int Pareja = 0; Pareja < Reemplazo; Pareja++)
        {
                //se eligen el padre y madre mediante torneo
                Padres [Pareja].Padre = Torneo (Individuos, Tamano_Poblacion);
                Padres [Pareja].Madre = Torneo (Individuos, Tamano_Poblacion);

                //el padre y madre deben ser diferentes
                while (Padres [Pareja].Padre == Padres [Pareja].Madre)
                {
                        Padres [Pareja].Madre = Torneo (Individuos, Tamano_Poblacion);
                }
        }

        return Padres;
}
```

Funcion **Cruzamiento**:

```
Cromosoma * Cruzamiento (Cromosoma * Individuos,
                         int          Tamano_Poblacion,
                         int          Porcentaje_Reemplazo,
                         int          Tamano_Cromosoma,
                         Pareja     * Padres,
                         int          Bloques_a_Intercalar)
{
        int          Reemplazo;
        int          Tamano_Bloques_de_Cruza;
        Cromosoma * Hijo;

        //se calcula el numero de individuos nuevos a crear para la sig generacion
        Reemplazo = (Porcentaje_Reemplazo * Tamano_Poblacion) / 100;
        Hijo      = Dimensiona_Poblacion (Reemplazo, Tamano_Cromosoma);

        //se calcula en numero de genes que transmitira cada "padre"
        Tamano_Bloques_de_Cruza = Tamano_Cromosoma / Bloques_a_Intercalar;

        //la cruza se hará entre parejas (arreglo "Padres" de tamaño Reemplazo)
        for (int Pareja = 0; Pareja < Reemplazo; Pareja++)
        {
```

```
            //se "heredarán" los genes intercalando los bloques de los padres
            //desde el inicio hasta el ultimo punto de cruza, sin llegar al final
            //del cromosoma
            for (int Punto_de_Cruza = 0;
                Punto_de_Cruza < (Bloques_a_Intercalar - 1);
                Punto_de_Cruza++)
            {
                //se "heredan" los genes desde un punto al otro, hasta el ultimo
                //punto de cruza
                for (int Gen = (Punto_de_Cruza * Tamano_Bloques_de_Cruza);
                    Gen < ((Punto_de_Cruza + 1) * Tamano_Bloques_de_Cruza);
                    Gen++)
                {
                    //si el bloque a heredar es impar
                    //se heredan los genes del "padre" de ese bloque
                    if ((Punto_de_Cruza % 2) == 0)
                    {
                        Hijo [Pareja].Gen [Gen] = Individuos [Padres
                        [Pareja].Padre].Gen [Gen];
                    }
                    //en caso contrario, se heredan los genes de la "madre"
                    else
                    {
                        Hijo [Pareja].Gen [Gen] = Individuos [Padres
                        [Pareja].Madre].Gen [Gen];
                    }
                }//fin for que recorre cada gen
            }//fin for de puntos de cruza-1

            //se recorre desde el ultimo punto de cruza, hasta el final del cromosoma
            for (int Gen = ((Bloques_a_Intercalar - 1) * Tamano_Bloques_de_Cruza);
                Gen < Tamano_Cromosoma;
                Gen++)
            {
                //si el bloque a heredar es impar (que comienza en un punto par), se
                //heredan los genes del "padre" de ese bloque
                if ((Bloques_a_Intercalar % 2) != 0)
                {
                    Hijo [Pareja].Gen [Gen] = Individuos [Padres [Pareja].Padre].Gen
                    [Gen];
                }
                //en caso contrario, se heredan los genes de la "madre"
                else
                {
                    Hijo [Pareja].Gen [Gen] = Individuos [Padres [Pareja].Madre].Gen
                    [Gen];
                }
            }//fin for de ultimo recorrido de genes
        }//fin for que recorre las parejas

        return Hijo;
}
```

Funcion **Reemplazo**:

```
Cromosoma * Reemplazo (Cromosoma * Individuos,
                       Cromosoma * Hijos,
                       int         Tamano_Poblacion,
                       int         Porcentaje_Reemplazo)
{
        int Reemplazo;

        //se calcula el numero de individuos a desechar en la generacion, para
        //cambiarlos por los nuevos
        Reemplazo = (Porcentaje_Reemplazo * Tamano_Poblacion) / 100;

        //se ordenan los individuos para saber quienes serán desechados
        Individuos = Ordena_Poblacion(Individuos, Tamano_Poblacion);

        for (int Individuo = 0; Individuo < Reemplazo; Individuo++)
        {
                Individuos [Individuo] = Hijos [Individuo];
        }

        return Individuos;
}
```

Funcion **Mutacion**:

```
Cromosoma * Mutacion(Cromosoma * Individuos,
                     int        Tamano_Poblacion,
                     int        Tamano_Cromosoma,
                     int        Indice_Mutacion)
{
   int Posicion;

   randomize ();

   for (int Individuo = 0; Individuo < Tamano_Poblacion; Individuo++)
   {
      //se le asigna una probabilidad de mutacion al individuo
      Individuos [Individuo].Probabilidad_Mutacion  = rand() % 101;

      //si la probabilidad de mutacion del individuo esa dentro del rango del indice
      //de mutacion, al individuo se le muta una posicion al azar de su cromosoma
      for (int Posiciones_a_Mutar = 0; Posiciones_a_Mutar < 3; Posiciones_a_Mutar++)
      {
            if (Individuos [Individuo].Probabilidad_Mutacion <  Indice_Mutacion)
            {
               Posicion = rand () % Tamano_Cromosoma;
               if (Individuos [Individuo].Gen [Posicion] == 1)
               {
                  Individuos [Individuo].Gen [Posicion] = 0;
               }
```

```
                else
                {
                   Individuos [Individuo].Gen [Posicion] = 1;
                }
             }//fin si el individuo se muta
       }
   }//fin de recorrido de individuos

   return Individuos;
}
```

Funcion **Ejecuta_AG**:

```
void Ejecuta_AG (Cromosoma      * & Poblacion,
                 THash_map_type *   HMPalabras,
                 String         *   Subcadenas,
                 int            *   Subcadenas_frec,
                 int                Prefijos,
                 int                Raices,
                 int                Sufijos,
                 int                Tamano_Poblacion,
                 THash_map_type *   HMPrefijos,
                 THash_map_type *   HMRaices,
                 THash_map_type *   HMSufijos,
                 int                ParamGeneraciones,
                 TApplication   *   App,
                 TLabel         *   lbl)
{
   int Indice_Mutacion;
   int Porcentaje_Reemplazo;
   int Bloques_a_Intercalar;
   int Generaciones;

   Cromosoma * Hijos = NULL;

   int Tamano_Cromosoma = Prefijos + Raices + Sufijos;

   //genera la poblacion inicial
   Poblacion = Genera_Poblacion_Inicial (Tamano_Poblacion, Tamano_Cromosoma);

   //Inicio de la primera etapa del AG
   //en esta etapa el algoritmo mejora la poblacion inicial
   //solo para hacerla llegar a un punto en el que las soluciones
   //no sean tan malas como al inicio
   Indice_Mutacion      = 30;
   Porcentaje_Reemplazo = 60;
   Bloques_a_Intercalar = 5;
   Generaciones         = ParamGeneraciones;

   for (int Generacion = 0; Generacion < Generaciones; Generacion++)
   {
      //evaluacion
```

```
    Evaluacion (Poblacion,
                Tamano_Poblacion,
                Tamano_Cromosoma,
                HMPalabras,
                Subcadenas,
                Prefijos,
                Raices,
                Sufijos,
                HMPrefijos,
                HMRaices,
                HMSufijos);
    //seleccion
    Pareja * Padres;
    Padres = Seleccion (Poblacion,
                        Tamano_Poblacion,
                    Porcentaje_Reemplazo);
    //cruzamiento
    Hijos  = Cruzamiento (Poblacion,
                      Tamano_Poblacion,
                      Porcentaje_Reemplazo,
                      Tamano_Cromosoma,
                      Padres,
                      Bloques_a_Intercalar);
    delete (Padres);
    //reemplazo
    Reemplazo (Poblacion,
               Hijos,
               Tamano_Poblacion,
               Porcentaje_Reemplazo);
     delete (Hijos);

    //mutacion
    Mutacion (Poblacion,
              Tamano_Poblacion,
              Tamano_Cromosoma,
              Indice_Mutacion);
    Indice_Mutacion -= Indice_Mutacion / Generaciones;

                lbl -> Caption = "1 etapa de 3, Generacion " + IntToStr (Generacion);
                App -> ProcessMessages ();
}//fin for de mejora genetica

//Segunda etapa del AG
//en esta se da una sacudida a las soluciones para
//ampliar el espacio de busqueda y no estancar el AG en un
//maximo local
Indice_Mutacion      = 80;
Porcentaje_Reemplazo = 80;
Bloques_a_Intercalar = 20;
Generaciones         = Generaciones * 2 /3;

for (int Generacion = 0; Generacion < Generaciones; Generacion++)
{
```

```
    //evaluacion
    Evaluacion (Poblacion,
                Tamano_Poblacion,
                Tamano_Cromosoma,
                HMPalabras,
                Subcadenas,
                Prefijos,
                Raices,
                Sufijos,
                HMPrefijos,
                HMRaices,
                HMSufijos);
    //seleccion
    Pareja *Padres;

    Padres = Seleccion (Poblacion,
                        Tamano_Poblacion,
                        Porcentaje_Reemplazo);
    //cruzamiento
    Hijos = Cruzamiento (Poblacion,
                         Tamano_Poblacion,
                         Porcentaje_Reemplazo,
                         Tamano_Cromosoma,
                         Padres,
                         Bloques_a_Intercalar);
     delete (Padres);
    //reemplazo
    Reemplazo (Poblacion,
               Hijos,
               Tamano_Poblacion,
               Porcentaje_Reemplazo);
     delete (Hijos);

    //mutacion
    Mutacion (Poblacion,
              Tamano_Poblacion,
              Tamano_Cromosoma,
              Indice_Mutacion);

                 Indice_Mutacion -= Indice_Mutacion / Generaciones;

                 lbl -> Caption = "2 etapa de 3, Generacion " + IntToStr (Generacion);
                 App -> ProcessMessages ();
}//fin for de mejora genetica

//Tercera etapa del AG
//estabiliza el AG de forma que las mejores soluciones encontradas
//no se pierda y se espera que gradualmente, las soluciones existentes
//se vayan mejorando
Indice_Mutacion      = 20;
Porcentaje_Reemplazo = 40;
Bloques_a_Intercalar = 4;
Generaciones         = Generaciones;
```

```
   for (int Generacion = 0; Generacion < Generaciones; Generacion++)
   {
      //evaluacion
      Evaluacion (Poblacion,
                                          Tamano_Poblacion,
                  Tamano_Cromosoma,
                  HMPalabras,
                  Subcadenas,
                  Prefijos,
                  Raices,
                  Sufijos,
                  HMPrefijos,
                  HMRaices,
                  HMSufijos);
      //seleccion
      Pareja * Padres;

      Padres = Seleccion (Poblacion,
                        Tamano_Poblacion,
                        Porcentaje_Reemplazo);
      //cruzamiento
      Hijos = Cruzamiento (Poblacion,
                         Tamano_Poblacion,
                         Porcentaje_Reemplazo,
                         Tamano_Cromosoma,
                         Padres,
                         Bloques_a_Intercalar);
       delete (Padres);
      //reemplazo
      Reemplazo (Poblacion,
                 Hijos,
                 Tamano_Poblacion,
                 Porcentaje_Reemplazo);
      delete (Hijos);
      //mutacion
      Mutacion (Poblacion,
                Tamano_Poblacion,
                Tamano_Cromosoma,
                Indice_Mutacion);

      Indice_Mutacion -= Indice_Mutacion / Generaciones;

      lbl -> Caption = "3 etapa de 3, Generacion " + IntToStr (Generacion);
                  App -> ProcessMessages ();
   }//fin for de mejora genetica

   //se ordenan los la poblacion final
   //Mejor individuo es el ultimo
   Ordena_Poblacion (Poblacion, Tamano_Poblacion);
}
```

ANEXO 2. Fragmento de la lista de palabras segmentada con el algoritmo genético

A continuación se muestran los resultados de las divisiones de algunas palabras realizadas con la información resultante del algoritmo para la lista completa. Es una de las posibles soluciones que se generaron.

Las rodeadas por el signo **=** son las raíces detectadas por el algoritmo. La subcadena a la derecha de la raíz es el sufijo de la palabra, de igual manera, en caso de no existir subcadena en esta parte es porque la palabra no posee un sufijo que haya sido detectado en el algoritmo.

Después de los signos **//** van otras posibles variantes de la división. Aunque se debe considerarse la primera como la variante correcta.

Recordamos que las palabras se presentan sin sus flexiones, por ejemplo, la raíz "atac-" corresponde a la palabra "atacar", "atad-" a la palabra "atado", etc.

=añ=
=añad=
=añadid=
=añadid=ur
=añaga=z // =añagaz=
=añej=
=añic=os // =añicos=
=añil=
=añor=
=añoranz=
=abad=
=abad=es // =abade=s
=abalan=z // =abalanz=
=abal=e // =abale=
=aband=er // =abander=
=aband=erad // =abander=ad // =abandera=d // =abanderad=
=abandera=mient // =abanderam=ient // =abanderamie=nt // =abanderamien=t
=aband=on
=aband=onism // =abandonism=
=aband=onist // =abandonist=
=abanic=
=abara=t
=abaratam=ient // =abaratami=ent // =abaratamie=nt
=abarc=
=abarro=t // =abarrot=
=abas=t
=abastec=
=abasteci=mient // =abastecimi=ent // =abastecimient=
=abat=
=abati=mient // =abatimie=nt // =abatimien=t
=abdi=c

=abdome=n
=abdomi=nal // =abdomina=l // =abdominal=
=abece=
=abeced=ari // =abecedari=
=abedul=
=abej=
=abej=on // =abejo=n // =abejon=
=abejor=r // =abejorr=
=aberr=acion // =aberra=cion // =aberrac=ion // =aberracion=
=abertur=
=abet=
=abier=t
=abigar=rad // =abigarr=ad // =abigarrad=
=abi=sm // =abis=m
=abjur=
=abland=
=ablu=cion // =abluci=on
=abnegac=ion // =abnegaci=on
=abne=gad
=aboc=
=aboca=d
=abochor=n
=abochor=nad // =abochornad=
=abof=ete
=abog=
=abogac=i
=abog=ad // =abogad=
=abol=
=abolen=g // =aboleng=
=abol=icion // =abolic=ion // =abolicio=n
=abolicio=nism // =abolicioni=sm // =abolicionism=
=abolicio=nist // =abolicioni=st
=abomb=
=abombad=
=abomin=
=abomina=bl // =abominab=l // =abominabl=
=abomina=cion // =abominaci=on // =abominacion=
=abon=
=abon=ad // =abona=d
=abona=mient // =abonam=ient // =abonami=ent // =abonamie=nt // =abonamien=t
=abord=
=abord=abl // =aborda=bl // =abordab=l
=abord=aje // =abordaj=e
=aborigen=
=aborre=c
=aborre=g
=abort=
=abr=
=abracadab=r // =abracadabr=
=abras=
=abras=iv // =abrasi=v
=abraz=
=abrazad=er // =abrazade=r
=abrelatas=
=abre=v
=abreva=der // =abrevader=
=abrevi=
=abrevi=ad
=abreviatur=
=abri=dor // =abridor=
=abri=g // =abrig=
=abri=l
=abrill=ant
=abrillanta=dor // =abrillantado=r // =abrillantador=
=abroch=
=abrog=
=abru=m

=abrumador=
=abrupt=
=absces=
=absolu=cion // =absoluc=ion // =absoluci=on // =absolucion=
=absolu=t // =absolut=
=absolu=tism // =absolut=ism // =absolutism=
=absolut=ist
=absol=v // =absolv=
=absorb=
=absorb=ent // =absorben=t
=absorb=ente // =absorben=te // =absorbente=
=absorcio=n
=absort=
=abstem=i // =abstemi=
=absten=
=absten=cion // =abstenc=ion // =abstenci=on // =abstencion=
=abstenci=onism // =abstencion=ism // =abstencioni=sm // =abstencionism=
=abstenci=onist // =abstencion=ist // =abstencioni=st // =abstencionist=
=abstinen=ci // =abstinenci=
=abstinen=t
=abstinen=te
=abstra=
=abstra=ccion // =abstrac=cion // =abstracci=on // =abstraccio=n
=abstra=ct // =abstrac=t
=abstru=s // =abstrus=
=absur=d
=absurdidad=
=abub=ill // =abubi=ll
=abuch=e // =abuche=
=abue=l // =abuel=
=abul=t
=abul=tad
=abund=
=abunda=nci
=abunda=nt // =abundant=
=aburgues=
=aburr=
=aburr=id // =aburrid=
=aburrimie=nt
=abus=
=abus=iv // =abusiv=
=abye=ct // =abyec=t
=aca=b // =acab=
=acab=ad
=acabo=se // =acabos=e // =acabose=
=aca=ci
=acade=mi // =academi=
=academi=c // =academic=
=academi=cism // =academic=ism // =academici=sm // =academicis=m
=aca=ec // =acaec=
=aca=ll
=acalo=r
=acalo=rad // =acalora=d
=acam=p // =acamp=
=acantil=ad // =acantilad=
=acanton=
=acap=ar
=acapara=dor // =acaparador=
=acari=ci // =acaric=i // =acarici=
=acar=re // =acarre=
=aca=t
=acatarr=
=acaud=al // =acauda=l
=acauda=lad // =acaudala=d
=acaud=ill // =acaudill=
=acced=
=acc=es // =acces=

=accesibili=dad // =accesibilid=ad // =accesibilida=d // =accesibilidad=
=acces=ibl // =accesi=bl // =accesib=l
=acces=ori
=accide=nt
=accidentad=
=accide=ntal
=accide=nte
=acc=ion // =acci=on // =accio=n
=acci=onist // =accio=nist // =accioni=st // =accionis=t
=aceñ=
=acech=
=acechan=z
=acei=t // =aceit=
=acei=te // =aceit=e // =aceite=
=acei=ter // =aceit=er // =aceite=r // =aceiter=
=acei=tos // =aceit=os // =aceitos=
=aceitu=n
=aceitu=ner
=aceler=
=acelera=cion // =acelerac=ion // =aceleraci=on // =aceleracio=n // =aceleracion=
=acelera=dor // =acelerad=or // =acelerado=r // =acelerador=
=aceleratri=z
=aceleron=
=acelg=
=acemil=
=acemiler=
=acend=r // =acendr=
=acend=rad // =acendr=ad // =acendra=d
=acent=
=acentu=
=acentuad=
=acep=cion // =acepc=ion // =acepci=on // =acepcion=
=acep=t
=aceptab=l // =aceptabl=
=aceptac=ion // =aceptaci=on // =aceptacion=
=acequi=
=acer=
=acer=ad // =acerad=
=acer=b
=acer=c
=acerca=mient // =acercamient=
=acerri=m
=acer=t // =acert=
=acer=tad // =acert=ad // =acerta=d
=acertij=
=acer=v // =acerv=
=acha=c // =achac=
=achac=os // =achacos=
=achapar=rad // =achaparr=ad
=achaq=ue // =achaqu=e
=ach=ic
=achicharr=
=achico=ri // =achicor=i // =achicori=
=achis=p
=achu=lad // =achulad=
=aciag=
=aciba=r
=acica=l
=acica=lad // =acicalad=
=acica=te // =acicate=
=aci=d
=acidez=
=aciert=
=aci=m
=acimu=t
=aclam=
=aclamaci=on // =aclamacion=

=aclar=
=aclar=acion // =aclara=cion
=aclimat=
=acn=e
=acobard=
=aco=d
=aco=dad
=aco=g
=acoged=or // =acogedo=r // =acogedor=
=acogid=
=acoli=t
=acom=et
=acometid=
=acomo=d // =acomod=
=acomo=dad // =acomod=ad // =acomoda=d // =acomodad=
=acomoda=diz // =acomodad=iz
=acomo=dador // =acomod=ador // =acomoda=dor // =acomodad=or // =acomodado=r // =acomodador=
=acomodati=ci // =acomodatici=
=acompañ=
=acompañ=ad // =acompaña=d // =acompañad=
=acompaña=mient // =acompañami=ent // =acompañamien=t
=acomp=as // =acompa=s
=acompa=sad // =acompasad=
=acond=icion // =acondi=cion // =acondici=on // =acondicion=
=acondicion=ador // =acondicionad=or
=acongoj=
=aconsej=
=aconsej=abl // =aconseja=bl // =aconsejabl=
=aconsej=ad // =aconseja=d // =aconsejad=
=acontec=
=acontecim=ient
=acop=i
=acop=l // =acopl=
=acopl=adur // =acopladu=r // =acopladur=
=acoplamie=nt // =acoplamien=t // =acoplamient=
=acop=le // =acopl=e // =acople=
=acorazad=
=acor=d
=acor=de // =acorde=
=acorde=on // =acordeo=n // =acordeon=
=acorde=onist // =acordeo=nist // =acordeon=ist // =acordeonis=t
=acordo=n // =acordon=
=acor=ral // =acorra=l
=acor=t
=aco=s // =acos=
=aco=st // =acos=t // =acost=
=acostumbr=
=acostumbr=ad // =acostumbra=d // =acostumbrad=
=aco=t
=acota=cion // =acotac=ion // =acotaci=on
=acr=
=acre=
=acre=c
=acrece=nt // =acrecent=
=acredi=t
=acre=edor // =acreed=or
=acrib=ill // =acribi=ll // =acribill=
=acrimoni=
=acrio=ll // =acrioll=
=acris=ol // =acriso=l
=acriso=lad // =acrisolad=
=acrit=ud
=acroba=ci // =acrobac=i // =acrobaci=
=acrob=at // =acroba=t
=acrob=atic // =acroba=tic
=act=
=actitu=d

=activ=
=actividad=
=activist=
=acto=r // =actor=
=actri=z // =actriz=
=actu=
=actuaci=on
=actual=
=actualida=d
=actualiz=
=actualiz=acion // =actualizacio=n
=acuñ=
=acuare=l
=acua=ri // =acuari=
=acuart=el // =acuarte=l // =acuartel=
=acuartelami=ent // =acuartelamie=nt // =acuartelamien=t // =acuartelamient=
=acua=tic // =acuat=ic // =acuati=c
=acuch=ill
=acuci=
=acuciant=
=acud=
=acuedu=ct // =acueduc=t
=acuer=d
=acumu=l // =acumul=
=acumul=acion // =acumulacio=n
=acun=
=acuos=
=acupuntur=
=acurruc=
=acus=
=acus=acion // =acusa=cion // =acusaci=on // =acusacio=n // =acusacion=
=acus=ad // =acusa=d // =acusad=
=acus=ador // =acusa=dor // =acusad=or // =acusado=r // =acusador=
=acus=ativ // =acusa=tiv // =acusat=iv // =acusativ=
=acus=atori // =acusa=tori // =acusat=ori // =acusato=ri // =acusator=i // =acusatori=
=acus=ic // =acusi=c // =acusic=
=acus=on
=acus=tic
=adag=i
=adal=id // =adali=d
=adan=
=adap=t
=adapta=cion // =adaptac=ion
=adecu=
=adecua=d // =adecuad=
=adefesi=
=adel=ant // =adelant=
=adelant=ad
=adel=f
=adelg=az // =adelgaz=
=adelgaza=mient // =adelgazam=ient // =adelgazami=ent // =adelgazamien=t // =adelgazamient=
=adema=n // =ademan=
=adent=r // =adentr=
=adept=
=ader=ez
=adestr=
=adeu=d
=adh=er // =adher=
=adher=enci // =adhere=nci // =adheren=ci // =adherenci=
=adher=ent // =adhere=nt // =adheren=t // =adherent=
=adhesi=on // =adhesio=n // =adhesion=
=adhesi=v
=adicion=
=adiciona=l
=adict=
=adiest=r // =adiestr=
=adine=rad // =adiner=ad // =adinera=d // =adinerad=

=adio=s
=adiv=in // =adivin=
=adivin=acion // =adivinac=ion // =adivinaci=on // =adivinacio=n
=adivinanz=
=adjeti=v
=adjud=ic // =adjudi=c // =adjudic=
=adju=nt // =adjun=t
=administ=r // =administr=
=administr=acion
=administ=rador // =administr=ador // =administrador=
=administ=rativ // =administr=ativ // =administrativ=
=admir=
=admir=abl // =admira=bl // =admirab=l
=admir=acion // =admira=cion
=admir=ador // =admira=dor // =admirado=r
=admis=ibl // =admisib=l
=admis=ion
=adm=it // =admit=
=admon=icion // =admonici=on // =admonicion=
=adobe=
=adoce=nad // =adocen=ad // =adocena=d
=adole=c // =adolec=
=adolesc=enci // =adolesce=nci // =adolescen=ci // =adolescenci=
=adolesc=ente // =adolesce=nte // =adolescen=te // =adolescent=e // =adolescente=
=adop=cion // =adopc=ion // =adopci=on // =adopcio=n
=adop=t
=adop=tiv // =adoptiv=
=adoqu=in // =adoqui=n
=adoqu=inad // =adoqui=nad // =adoquina=d
=ador=
=ador=abl // =adora=bl // =adorab=l
=ador=acion // =adora=cion // =adorac=ion // =adoracion=
=adorm=ec // =adorme=c // =adormec=
=adormi=der // =adormide=r
=ador=n // =adorn=
=ador=nist // =adorn=ist
=ados=
=adquir=
=adquis=icion // =adquisici=on // =adquisicio=n
=adquisiti=v // =adquisitiv=
=adriati=c
=adu=an // =adua=n
=adua=ner // =aduaner=
=adu=c // =aduc=
=adu=eñ // =adueñ=
=adu=l
=adula=cion // =adulac=ion // =adulacion=
=adula=dor // =adulad=or
=adulo=n // =adulon=
=adult=
=adult=er // =adulte=r
=adult=eri // =adulte=ri // =adulteri=
=adu=st // =adus=t // =adust=
=adve=n // =adven=
=advene=diz // =advened=iz // =advenedi=z // =advenediz=
=adveni=mient // =advenim=ient // =advenimien=t
=adventi=ci // =adventici=
=adverbi=
=adverbia=l // =adverbial=
=adver=s // =advers=
=advers=ari // =adversar=i
=adver=sidad // =advers=idad // =adversi=dad // =adversida=d
=adver=t // =advert=
=adver=tenci // =advert=enci // =adverten=ci // =advertenc=i // =advertenci=
=adver=tid // =advert=id
=adyacent=
=aer=e

=aerodesliz=ador // =aerodesliza=dor
=aerodinam=ic // =aerodinami=c
=aerodrom=
=aeromoz=
=aeronaut=
=aeronaut=ic // =aeronauti=c
=aeronav=e // =aeronave=
=aeropuer=t
=aerosol=
=aerospa=cial // =aerospaci=al // =aerospacia=l
=aerosta=t
=aerosta=tic // =aerostati=c // =aerostatic=
=aerotransport=ad // =aerotransporta=d // =aerotransportad=
=afab=l // =afabl=
=afam=ad
=afan=
=afe=
=afec=cion
=afec=t
=afecta=cion // =afectacion=
=afec=tad // =afecta=d
=afectisim=
=afec=tiv // =afectiv=
=afectuos=
=afei=t
=afei=te
=afem=inad // =afemi=nad // =afemina=d // =afeminad=
=afer=r // =aferr=
=afga=n // =afgan=
=afian=z
=afici=on // =aficio=n // =aficion=
=afici=onad // =aficio=nad // =aficion=ad
=afij=
=afil=
=afil=ad // =afila=d // =afilad=
=afil=ador // =afila=dor // =afilad=or
=afil=adur // =afila=dur // =afilad=ur // =afiladu=r // =afiladur=
=afilalapic=es // =afilalapice=s
=afila=mient // =afilami=ent // =afilamie=nt // =afilamient=
=afil=i // =afili=
=afili=acion // =afilia=cion // =afiliac=ion // =afiliaci=on // =afiliacion=
=afiligra=nad // =afiligranad=
=afin=
=afina=dor // =afinador=
=afinc=
=afini=dad // =afinidad=
=afirm=
=afirm=acion // =afirma=cion // =afirmac=ion
=afirm=ativ // =afirma=tiv // =afirmativ=
=aflic=cion // =aflicc=ion
=aflig=
=afligi=d // =afligid=
=afloj=
=aflo=r
=afloramie=nt // =afloramient=
=aflu=
=aflu=enci // =afluen=ci // =afluenc=i // =afluenci=
=aflu=ent // =afluen=t // =afluent=
=aflu=ente // =afluen=te // =afluent=e
=aforis=m // =aforism=
=afortun=ad
=afrances=ad // =afrancesa=d // =afrancesad=
=afr=ent // =afre=nt // =afren=t
=afric=an // =africa=n // =african=
=afric=anism // =africa=nism // =african=ism // =africani=sm // =africanis=m
=africa=nist // =african=ist // =africani=st // =africanis=t
=afroasi=atic

=afrocuban=
=afront=
=afu=ste // =afuste=
=agüe=r
=agach=
=agall=
=agar=r
=agar=rad
=agarraderas=
=agar=ron // =agarron=
=agarrot=
=agasaj=
=agat=
=agav=ill // =agavi=ll // =agavill=
=agazap=
=agenci=
=agend=
=agent=
=agente=
=agi=l
=agi=lidad // =agilid=ad // =agilida=d
=agi=liz // =agiliz=
=agi=t // =agit=
=agit=acion // =agitacio=n // =agitacion=
=agi=tador // =agit=ador
=aglom=er // =aglomer=
=aglomer=acion // =aglomera=cion
=aglutin=
=agob=i // =agobi=
=agol=p // =agolp=
=ago=ni // =agon=i // =agoni=
=agon=iz // =agoni=z // =agoniz=
=ago=r
=agorer=
=ago=st
=ago=t
=ago=tador
=agr=
=agra=ci
=agraciad=
=agr=ad // =agra=d // =agrad=
=agrad=abl // =agrada=bl // =agradab=l
=agrad=ec // =agrade=c
=agrade=cid // =agradeci=d // =agradecid=
=agradeci=mient // =agradecimien=t
=agran=d
=agr=ari // =agra=ri // =agrar=i
=agra=v
=agrava=cion
=agrava=mient
=agrava=nt
=agrava=nte
=agravi=
=agravi=ad // =agravia=d
=agred=
=agreg=
=agreg=ad
=agresio=n
=agresiv=
=agresivida=d // =agresividad=
=agreso=r
=agrest=
=agr=i
=agrico=l
=agricu=ltor // =agricul=tor
=agricu=ltur // =agricul=tur // =agricultu=r // =agricultur=
=agriet=

=agronom=
=agronom=i // =agronomi=
=agronom=ic // =agronomi=c
=agropecu=ari // =agropecua=ri // =agropecuari=
=agrup=
=agrupa=cion // =agrupac=ion // =agrupacio=n
=agrupa=mient // =agrupam=ient // =agrupami=ent // =agrupamie=nt
=agu=
=aguacate=
=aguac=er // =aguace=r // =aguacer=
=aguac=il
=aguafiest=as
=aguafuer=te // =aguafuert=e
=aguaman=il // =aguamani=l
=aguama=nos // =aguaman=os // =aguamano=s
=aguaniev=e
=aguaniev=es
=agua=nt // =aguan=t // =aguant=
=agua=nte // =aguan=te // =aguant=e // =aguante=
=aguard=
=aguard=iente // =aguardien=te // =aguardiente=
=agud=
=agud=ez // =agude=z // =agudez=
=agud=iz // =agudiz=
=aguerr=id
=aguijad=
=aguijon=
=aguijone=
=agui=l
=aguileñ=
=aguinald=
=aguj=
=aguje=r // =agujer=
=aguje=re // =agujer=e
=aguje=t
=aguz=
=aguzaniev=es // =aguzanieve=s // =aguzanieves=
=ahin=c // =ahinc=
=aho=g
=aho=gad // =ahoga=d
=ahon=d
=ahor=c
=ahor=r // =ahorr=
=ahor=rativ // =ahorr=ativ // =ahorrat=iv
=ahu=ec // =ahue=c // =ahuec=
=ahu=m
=ahuye=nt // =ahuyen=t
=airad=
=aire=
=airos=
=ais=l
=aislaci=onism // =aislacionis=m // =aislacionism=
=aislaci=onist // =aislacionis=t
=ais=lad // =aisla=d
=aisla=dor // =aislado=r // =aislador=
=aisla=mient // =aislam=ient // =aislami=ent
=aj=
=ajedrecist=
=ajedr=ez
=aje=n
=ajenj=
=ajetr=e // =ajetre=
=aji=
=ajiac=
=aju=ar
=aju=st // =ajust=
=ajust=ad // =ajustad=

=ajust=ador // =ajustad=or // =ajustado=r
=aju=ste // =ajust=e // =ajuste=
=ajustici=
=ajusticiami=ent // =ajusticiamie=nt // =ajusticiamien=t // =ajusticiamient=
=al=
=ala=b
=alaban=z // =alabanz=
=alabar=d // =alabard=
=alabar=der // =alabard=er
=alabastr=
=alacen=
=alacra=n // =alacran=
=ala=d
=ala=m // =alam=
=alamb=ic // =alambi=c
=alamb=icad // =alambi=cad // =alambicad=
=alambique=
=alamb=rad // =alambra=d
=alamb=re // =alambre=
=alame=d // =alamed=
=alar=de // =alarde=
=alar=g
=alar=id // =alari=d
=alar=m // =alarm=
=alarm=ant // =alarman=t
=alarm=ist // =alarmi=st // =alarmis=t // =alarmist=
=alaza=n
=alb=
=albañ=il // =albañi=l // =albañil=
=albañil=eri // =albañile=ri // =albañileri=
=albace=
=albaha=c
=alban=es
=albard=
=albaricoq=ue
=albaricoquer=
=albat=ros // =albatr=os
=albed=ri // =albedr=i // =albedri=
=alberg=
=albergue=
=albondig=
=alb=or // =albor=
=albor=ad // =alborad=
=albor=e
=albornoz=
=alboro=t
=alboro=tad // =alborota=d
=alboro=tador // =alborota=dor // =alborotado=r
=alboro=z
=albufer=
=albu=m // =album=
=alcachof=
=alcahu=et // =alcahue=t
=alcahu=ete // =alcahue=te // =alcahuete=
=alcal=dad // =alcald=ad // =alcaldad=
=alcal=de // =alcald=e
=alcal=di // =alcald=i
=alca=li // =alcal=i // =alcali=
=alcal=in // =alcali=n // =alcalin=
=alcance=
=alcanfo=r
=alcantarill=
=alcantarill=ad // =alcantarilla=d
=alcanz=
=alcaza=r
=alce=
=alco=b

=alcoho=l
=alcoholic=
=alcoho=lism // =alcoholism=
=alcoho=liz
=alcornoq=ue
=alcur=ni
=alda=b // =aldab=
=ald=e
=aldea=n
=ale=acion // =aleac=ion // =aleaci=on // =aleacion=
=ale=ccion // =alec=cion // =alecci=on
=alecciona=dor // =aleccionado=r
=aledañ=os // =aledaño=s // =aledaños=
=ale=g // =aleg=
=aleg=at // =alega=t // =alegat=
=aleg=ori // =alegor=i // =alegori=
=aleg=oric // =alegor=ic // =alegori=c // =alegoric=
=alegor=iz // =alegori=z // =alegoriz=
=aleg=r
=aleg=ri // =alegri=
=alej=
=aleluy=
=aleman=
=ale=nt // =alent=
=alent=ador // =alenta=dor // =alentad=or // =alentado=r // =alentador=
=ale=r
=alerce=
=alerge=n // =alergen=
=alergi=
=alergi=c // =alergic=
=alert=
=ale=t
=alev=in
=alev=os // =alevo=s // =alevos=
=alevos=i // =alevosi=
=alf=
=alfab=et
=alfabeti=c // =alfabetic=
=alfabeti=z // =alfabetiz=
=alfabetiz=acion // =alfabetizaci=on // =alfabetizacion=
=alfal=f // =alfalf=
=alfam=ar // =alfamar=
=alfar=er // =alfare=r
=alfar=eri // =alfare=ri
=alfeñiq=ue
=alfeiza=r // =alfeizar=
=alfer=ez // =alfere=z
=alf=il
=alfiler=
=alfombr=
=alfoncig=
=alforfo=n
=alforj=
=alg=
=algarabi=
=algaza=r // =algazar=
=algeb=r // =algebr=
=algebrai=c
=algeb=ric // =algebr=ic // =algebri=c // =algebric=
=algodon=
=algodon=al // =algodona=l // =algodonal=
=algodon=er // =algodone=r // =algodoner=
=alguac=il
=alguie=n // =alguien=
=alhaj=
=alhe=li // =alhel=i
=alhe=ti

=ali=
=aliñ=
=ali=ad
=alian=z // =alianz=
=ali=as // =alias=
=alicat=es // =alicate=s // =alicates=
=alic=iente // =alicien=te
=alien=
=alien=acion // =alienaci=on // =alienacion=
=ali=ent // =alien=t // =alient=
=alig=er
=alij=
=alimañ=
=ali=ment // =alim=ent // =alime=nt // =alimen=t // =aliment=
=aliment=acion // =alimentac=ion
=alime=ntari // =alimen=tari // =aliment=ari // =alimentar=i
=alimenti=ci
=ali=ne
=ali=s
=ali=st
=alista=mient // =alistamie=nt // =alistamien=t // =alistamient=
=alivi=
=alja=b
=aljibe=
=all=an // =allan=
=allana=mient // =allanami=ent // =allanamie=nt // =allanamient=
=alleg=
=alleg=ad // =allega=d
=alm=
=almacen=
=almacen=aje
=almacen=ist // =almaceni=st
=almanaq=ue // =almanaqu=e
=almej=
=almen=
=almend=r // =almendr=
=almib=ar // =almiba=r
=almiba=rad // =almibara=d
=almid=on // =almidon=
=almid=onad // =almidon=ad // =almidonad=
=almirantazg=
=almiran=te // =almirant=e
=almizcl=e
=almizcl=eñ // =almizcleñ=
=almizcl=er // =almizcler=
=almodro=te
=almoh=ad // =almoha=d
=almohadill=
=almoh=az // =almoha=z // =almohaz=
=almone=d // =almoned=
=almorz=
=almuerz=
=alocad=
=alocu=cion // =alocuc=ion // =alocucio=n
=aloe=
=aloj=
=alojamie=nt // =alojamient=
=alond=r
=alpa=c // =alpac=
=alpargat=
=alpin=
=alpinis=m
=alpinis=t
=alqu=eri // =alque=ri
=alqu=il
=alquile=r // =alquiler=
=alquim=i

=alquim=ist // =alquimis=t
=alquitran=
=alred=edor // =alrede=dor // =alreded=or // =alrededo=r
=alt=
=altaner=
=altaner=i
=alt=ar
=altavo=z // =altavoz=
=alt=er // =alte=r
=alteraci=on // =alteracio=n // =alteracion=
=alterc=
=alterc=ad // =altercad=
=altern=
=altern=ativ // =alternat=iv // =alternati=v // =alternativ=
=altern=e // =alterne=
=alt=ez // =alte=z // =altez=
=altibajos=
=altipla=n
=altiplanici=e // =altiplanicie=
=altisim=
=altis=on // =altiso=n // =altison=
=altiso=nant // =altison=ant // =altisona=nt // =altisonant=
=alt=itud // =altit=ud // =altitud=
=alt=iv // =alti=v // =altiv=
=altiv=ez // =altive=z // =altivez=
=altoparla=nte
=altruist=
=alt=ur // =altu=r // =altur=
=alub=i
=aluc=in // =aluci=n // =alucin=
=alucin=acion
=alud=
=alumb=r // =alumbr=
=alumb=rad // =alumbr=ad
=alumbram=ient // =alumbrami=ent // =alumbramie=nt // =alumbramien=t
=alumin=i // =alumini=
=alumn=
=alumn=ad
=aluni=z
=alunizaj=e
=alusi=on // =alusio=n // =alusion=
=alz=
=alzad=
=alzam=ient // =alzamie=nt // =alzamien=t
=am=
=amañ=
=amabi=lidad // =amabili=dad
=ama=bl // =amab=l
=ama=d // =amad=
=amaest=r
=ama=g
=ama=in // =amai=n
=amalgam=
=amance=b // =amanceb=
=aman=ec // =amane=c
=amanece=r
=aman=erad // =amane=rad // =amanerad=
=ama=ns // =aman=s
=ama=nt // =aman=t // =amant=
=ama=nte // =aman=te // =amant=e
=amanue=nse // =amanuens=e
=amapol=
=amarg=
=amarg=or
=amarg=ur
=amari=ll // =amarill=
=amarr=

=ama=s
=amateu=r // =amateur=
=ambage=s
=amba=r // =ambar=
=ambi=cion // =ambici=on // =ambicio=n // =ambicion=
=ambici=os // =ambicio=s
=ambi=ent // =ambie=nt // =ambien=t // =ambient=
=ambi=ental // =ambie=ntal // =ambien=tal // =ambient=al // =ambienta=l // =ambiental=
=ambi=ente // =ambie=nte // =ambien=te // =ambient=e
=ambigü=edad // =ambigüedad=
=ambigu=
=ambi=t // =ambit=
=ambival=ent // =ambivale=nt // =ambivalen=t
=ambo=n // =ambon=
=ambo=s
=ambros=i // =ambrosi=
=ambul=anci // =ambula=nci // =ambulan=ci // =ambulanc=i // =ambulanci=
=ambul=ant // =ambula=nt // =ambulan=t
=ambul=atori // =ambula=tori // =ambulat=ori // =ambulato=ri // =ambulator=i // =ambulatori=
=ame=b
=amedr=ent // =amedre=nt
=ame=n
=amena=z // =amenaz=
=amena=zador // =amenaz=ador // =amenazad=or // =amenazado=r // =amenazador=
=amenaz=ant // =amenazant=
=ameni=z // =ameniz=
=americ=an // =american=
=americ=anism // =american=ism
=ametralla=dor // =ametrallado=r
=amia=nt
=amib=
=amig=
=amig=abl // =amigab=l
=amigda=l
=amigui=sm // =amiguism=
=aminor=
=amis=tad // =amistad=
=amis=tos // =amisto=s // =amistos=
=amnis=ti // =amnist=i // =amnisti=
=amol=d // =amold=
=amonest=
=amonia=c
=amonto=n // =amonton=
=amo=r
=amo=ral // =amora=l
=amora=t // =amorat=
=amora=tad // =amorat=ad
=amorda=z // =amordaz=
=amorf=
=amo=ri // =amori=
=amo=ros // =amoro=s
=amortaj=
=amortigu=
=amortiguacion=
=amortiguador=
=amortiz=
=amortiza=cion // =amortizaci=on
=amosc=
=amo=tin // =amot=in // =amoti=n // =amotin=
=amot=inad // =amoti=nad // =amotin=ad // =amotinad=
=amp=ar // =ampar=
=amp=li // =ampl=i // =ampli=
=ampli=fic
=amplificad=or // =amplificado=r
=ampl=itud // =amplit=ud // =amplitu=d // =amplitud=
=ampo=ll
=ampu=los // =ampulo=s // =ampulos=

=ampu=t // =amput=
=amue=bl
=amul=et
=an=
=anacor=et // =anacoret=
=anacron=ic // =anacroni=c // =anacronic=
=anacr=onism // =anacron=ism // =anacroni=sm // =anacronis=m
=ana=l // =anal=
=anal=es // =anale=s // =anales=
=analfab=et // =analfabe=t // =analfabet=
=analfabe=tism // =analfabet=ism
=analisi=s
=ana=list // =anal=ist // =analis=t // =analist=
=analit=ic // =analiti=c // =analitic=
=ana=liz // =anal=iz
=ana=log // =anal=og // =analo=g
=ana=logi // =anal=ogi // =analo=gi // =analogi=
=anana=
=ananas=
=anaqu=el // =anaquel=
=anarqu=i
=anarqu=ic // =anarquic=
=anarqu=ist // =anarquis=t
=anatem=
=anato=mi
=anatomic=
=anatomis=t
=anc=
=ancestr=al // =ancestral=
=anch=
=ancho=
=anch=ur // =anchu=r // =anchur=
=anc=ian // =anci=an // =ancian=
=anc=l
=anc=or // =ancor=
=and=
=anda=d // =andad=
=andad=as // =andadas=
=anda=dur // =andad=ur // =andadu=r // =andadur=
=andalu=z // =andaluz=
=anda=mi // =andam=i // =andami=
=andam=iad // =andami=ad // =andamia=d
=andami=aje
=anda=nt // =andan=t // =andant=
=anda=nte // =andan=te // =andant=e
=andan=z // =andanz=
=anda=r // =andar=
=andarieg=
=anda=s
=ande=n
=andi=n // =andin=
=andraj=
=andrajos=
=andur=rial // =andurr=ial
=anecdot=
=anecdot=ic
=aneg=
=anej=
=anex=
=anex=ion // =anexi=on // =anexion=
=anfib=i
=anfiteat=r
=anfitrio=n // =anfitrion=
=anfo=r
=ange=l // =angel=
=angelic=al
=angi=n

=anglosajon=
=angost=
=anguil=
=angul=
=angul=ar
=angul=os
=angus=ti // =angusti=
=angusti=ad // =angustia=d
=anh=el
=anid=
=anill=
=anim=
=anim=acion // =animaci=on // =animacio=n
=anim=ad // =animad=
=anim=ador // =animad=or
=animadver=sion // =animadvers=ion
=anim=al
=anim=aliz // =animaliz=
=anim=ic // =animi=c // =animic=
=anim=os
=aniqu=il // =aniquil=
=aniquil=acion // =aniquila=cion // =aniquilac=ion // =aniquilaci=on
=aniquil=ad // =aniquila=d // =aniquilad=
=aniquila=mient // =aniquilam=ient // =aniquilamient=
=anis=
=anivers=ari
=anoche=c
=anochece=r // =anochecer=
=anod=in
=anom=al
=anomali=
=anonad=
=anonim=
=anonim=at // =anonimat=
=anorak=
=anorma=l
=anot=
=anotaci=on
=ansi=
=ansieda=d // =ansiedad=
=ansio=s
=antagon=ic // =antagoni=c // =antagonic=
=antagon=ism // =antagoni=sm // =antagonis=m
=antar=tic // =antarti=c
=antebraz=
=antecam=ar // =antecamar=
=anteced=
=antecede=nci // =anteceden=ci // =antecedenci=
=antecede=nt // =anteceden=t
=antecede=nte // =anteceden=te // =antecedente=
=antecesor=
=antedi=ch
=antediluvia=n // =antediluvian=
=antela=cion // =antelacion=
=ante=n
=anteoj=
=antep=on // =antepon=
=anteproyect=
=anteri=or // =anterio=r
=anterio=ridad // =anteriorid=ad // =anteriorida=d
=antes=al // =antesa=l
=antiaere=
=antiatomic=
=antiba=l
=antibiotic=
=anticip=
=anticipa=cion // =anticipaci=on // =anticipacion=

=anticipa=d // =anticipad=
=anticoncep=tiv // =anticoncept=iv // =anticoncepti=v // =anticonceptiv=
=anticongel=ador // =anticongelado=r // =anticongelador=
=anticu=ad // =anticua=d // =anticuad=
=anticu=ari // =anticua=ri // =anticuar=i
=antido=t
=antifasc=ist // =antifascist=
=antifaz=
=antigüedad=
=antig=as // =antigas=
=antigu=
=antiguall=
=antill=an // =antilla=n
=antilop=e
=antimo=ni // =antimon=i // =antimoni=
=antinatura=l // =antinatural=
=antipa=ti
=antipa=tic // =antipatic=
=antipatriot=
=antipatriot=ic // =antipatrioti=c
=antipire=tic // =antipiret=ic // =antipireti=c
=antipod=
=antisem=it // =antisemi=t // =antisemit=
=antisemi=tic // =antisemit=ic // =antisemiti=c
=antisemi=tism // =antisemit=ism // =antisemiti=sm
=antisoci=al
=antitanqu=
=antitesi=s // =antitesis=
=antoj=
=antoja=diz // =antojadiz=
=anto=logi // =antol=ogi // =antolo=gi // =antolog=i
=antonim=
=antonimi=
=antor=ch
=antr=
=anua=l
=anua=ri // =anuar=i
=anud=
=anul=
=anun=ci // =anunci=
=anunci=acion // =anuncia=cion // =anunciac=ion // =anunciacio=n // =anunciacion=
=anver=s
=anzue=l
=apañ=
=apañ=ad // =apaña=d
=apañ=ador // =apaña=dor // =apañador=
=apabull=
=apac=ibl // =apacib=l
=apacigu=
=apadr=in // =apadri=n
=apa=g
=apa=gad
=apago=n // =apagon=
=apa=le // =apal=e
=apa=rador // =apar=ador // =aparado=r // =aparador=
=apar=at // =aparat=
=aparat=os // =aparatos=
=apar=c // =aparc=
=aparca=mient // =aparcamie=nt // =aparcamien=t
=aparc=er
=aparc=eri // =aparceri=
=apar=ec // =aparec=
=aparec=id
=aparej=
=aparej=ad // =apareja=d
=apar=ent
=aparien=ci // =aparienc=i // =aparienci=

=apar=t // =apart=
=apar=tad // =apart=ad
=apartam=ent
=aparthei=d // =apartheid=
=apa=sion // =apas=ion // =apasi=on // =apasion=
=apas=ionad // =apasi=onad // =apasion=ad // =apasionad=
=apasion=ant // =apasionan=t // =apasionant=
=apa=ti
=apa=tic // =apatic=
=apatr=id
=ape=
=apedre=
=ape=g // =apeg=
=ape=l
=apelacio=n
=apelativ=
=apell=id // =apelli=d // =apellid=
=apelm=az // =apelma=z // =apelmaz=
=apeloto=n
=ape=n
=apenc=
=apendic=e // =apendice=
=apendicitis=
=ape=r
=aperci=b
=aperi=tiv // =aperiti=v // =aperitiv=
=apert=ur // =apertu=r
=apertu=rism // =aperturi=sm // =aperturis=m
=aperturi=st // =aperturis=t
=apesadumbr=
=ape=st
=apet=ec
=ape=tenci // =apet=enci // =apeten=ci // =apetenc=i // =apetenci=
=apet=it // =apeti=t // =apetit=
=apeti=tos // =apetit=os // =apetito=s // =apetitos=
=api=
=apiñ=
=apice=
=apicult=or // =apiculto=r // =apicultor=
=apicult=ur // =apicultur=
=api=l
=apison=
=apisonador=
=apla=c
=apl=an // =apla=n
=apla=st // =aplas=t // =aplast=
=aplast=ant // =aplasta=nt // =aplastan=t // =aplastant=
=apla=ud // =aplau=d // =aplaud=
=aplau=s // =aplaus=
=apl=az // =apla=z // =aplaz=
=apl=ic
=aplicab=l // =aplicabl=
=aplicacio=n
=apl=icad
=aplo=m
=apoc=
=apoc=ad // =apoca=d
=apocri=f
=apod=
=apod=er
=apod=erad
=apoge=
=apoli=tic // =apolit=ic // =apoliti=c
=apol=og // =apolo=g
=apol=ogi // =apolo=gi // =apologi=
=apologi=st // =apologis=t // =apologist=
=apor=re // =aporr=e // =aporre=

=apor=t // =aport=
=aport=acion // =aportacion=
=apor=te // =aport=e // =aporte=
=apos=ent // =apose=nt // =aposen=t // =aposent=
=apos=icion // =aposi=cion // =aposic=ion // =aposicio=n // =aposicion=
=apos=t
=apostas=i // =apostasi=
=aposta=t
=aposto=l // =apostol=
=aposto=lad // =apostol=ad // =apostolad=
=apostol=ic // =apostoli=c
=apostrof=
=apostrofe=
=apoteosis=
=apoy=
=apre=ci
=apreciabl=
=apreciacio=n
=aprehend=
=aprehensi=on // =aprehensio=n
=apre=mi // =aprem=i // =apremi=
=apremi=ant // =apremia=nt // =apremian=t
=apren=d // =aprend=
=apren=diz // =aprend=iz // =aprendiz=
=aprendiz=aje // =aprendizaj=e
=apren=sion // =aprens=ion // =aprensio=n // =aprension=
=aprens=iv
=apre=s // =apres=
=apres=ur
=apresurad=
=apre=t // =apret=
=apre=tad // =apret=ad // =apretad=
=apret=on // =apreton=
=apri=et // =aprie=t
=apri=sion // =apris=ion // =aprisi=on // =aprisio=n
=aprob=
=aprobac=ion // =aprobacion=
=aprobad=
=apront=
=apropi=
=apropi=acion // =apropiac=ion // =apropiaci=on
=apropi=ad
=aprove=ch
=aprove=chad // =aprovecha=d // =aprovechad=
=aprovecha=mient // =aprovechamie=nt // =aprovechamien=t // =aprovechamient=
=aprovisi=on // =aprovisio=n
=aproxim=
=aproximaci=on // =aproximacion=
=aproximad=
=apt=
=apt=itud // =aptit=ud // =aptitu=d
=apuña=l
=apue=st // =apuest=
=apun=t
=apun=tador // =apunta=dor // =apuntado=r // =apuntador=
=apun=tal // =apunta=l
=apun=te
=apur=
=apurad=
=apuro=s
=aquej=
=aquelarre=
=aquiet=
=aquila=t // =aquilat=
=aquilin=
=aquilon=
=ar=

=arañ=
=araña=z
=arab=
=arabe=
=arabes=c
=arabig=
=arabism=
=arabist=
=arad=
=aragone=s // =aragones=
=aranc=el // =arance=l
=arance=lari // =arancelar=i // =arancelari=
=arandel=
=arbit=r
=arbitraj=e
=arbitra=ri
=arbitrarie=dad // =arbitraried=ad
=arbit=ri // =arbitri=
=arbitri=st // =arbitrist=
=arb=ol // =arbo=l // =arbol=
=arboled=
=arbo=re // =arbore=
=arbu=st // =arbus=t // =arbust=
=arc=
=arcabuce=r
=arcabu=z
=arca=d // =arcad=
=arca=ic // =arcai=c
=arca=ism // =arcai=sm // =arcaism=
=arca=n // =arcan=
=arcangel=
=arce=
=arcedia=n // =arcedian=
=archidiac=on // =archidiacon=
=archidiocesis=
=archimandr=it // =archimandrit=
=archipampa=n
=archipiela=g // =archipielag=
=arch=iv
=archiver=
=arci=ll // =arcill=
=arci=llos // =arcill=os // =arcillo=s
=arcipres=te
=ard=
=ardid=
=ardid=ez // =ardide=z
=ardie=nt // =ardient=
=ardill=
=ardo=r // =ardor=
=ardo=ros // =ardor=os // =ardoro=s
=ardu=
=are=
=are=n
=are=nal // =arena=l // =arenal=
=areng=
=are=nos // =areno=s // =arenos=
=arenque=
=are=ol // =areo=l
=are=te // =arete=
=argü=
=argama=s // =argamas=
=argelin=
=argen=tin // =argent=in // =argenti=n
=argenti=nism // =argentini=sm // =argentinis=m
=argoll=
=arg=on
=argot=

=argu=ci
=argu=ment // =argum=ent // =argumen=t
=argumentaci=on // =argumentacion=
=ari=
=ari=d // =arid=
=arid=ez // =aride=z
=ari=es // =aries=
=ari=ete // =ariet=e // =ariete=
=arisc=
=ari=st // =arist=
=aristocra=ci // =aristocrac=i
=aristocra=t // =aristocrat=
=aristocra=tic // =aristocrat=ic // =aristocratic=
=aritme=tic // =aritmet=ic
=arlequ=in // =arlequi=n // =arlequin=
=arm=
=arm=ad // =arma=d // =armad=
=armad=ill
=arm=ador // =arma=dor // =armad=or // =armado=r // =armador=
=arm=adur // =arma=dur // =armad=ur // =armadur=
=arma=ment // =armame=nt // =armament=
=armament=ism // =armamenti=sm // =armamentism=
=armament=ist // =armamenti=st
=arm=ari // =arma=ri // =armari=
=armazo=n
=arme=ni // =armeni=
=arm=er // =arme=r // =armer=
=arm=eri // =arme=ri // =armer=i // =armeri=
=armiñ=
=armistic=i // =armistici=
=armon=i // =armoni=
=armon=ic // =armoni=c
=armon=ios // =armoni=os // =armonios=
=armon=iz // =armoni=z // =armoniz=
=arnic=
=arom=
=arom=atic // =aroma=tic // =aromatic=
=arom=atiz // =aroma=tiz // =aromatiz=
=arp=
=arp=i
=arp=on
=arqueolo=g // =arqueolog=
=arqueolo=gi // =arqueolog=i
=arqueti=p // =arquetip=
=arquite=ct // =arquitec=t
=arquitecton=ic // =arquitectonic=
=arquitec=tur
=arrab=al // =arrabal=
=arrai=g // =arraig=
=arrai=gad // =arraig=ad
=arramb=l // =arrambl=
=arran=c // =arranc=
=arranq=ue // =arranqu=e
=arr=as
=arrastr=
=arrastr=ad
=arrastr=e
=arr=e
=arrebat=
=arrebat=ad
=arrebol=
=arrechuch=
=arreci=
=arrecif=e // =arrecife=
=arregl=
=arregl=ad // =arregla=d // =arreglad=
=arregl=ist // =arregli=st // =arreglis=t // =arreglist=

=arreman=g // =arremang=
=arremet=
=arremet=id
=arremoli=n // =arremolin=
=arren=d // =arrend=
=arren=dador // =arrend=ador // =arrenda=dor
=arrenda=mient // =arrendam=ient // =arrendamie=nt // =arrendamien=t // =arrendamient=
=arrenda=tari // =arrendata=ri // =arrendatar=i
=arrepe=nt
=arrepenti=mient // =arrepentimi=ent // =arrepentimient=
=arres=t // =arrest=
=arr=i
=arrib=
=arrib=ism // =arribis=m
=arrib=ist // =arribis=t // =arribist=
=arrien=d // =arriend=
=arriesg=
=arriesg=ad // =arriesga=d
=arrim=
=arrinco=n
=arrinco=nad // =arrinconad=
=arris=cad // =arrisc=ad // =arrisca=d
=arro=b
=arrod=ill
=arr=og // =arro=g // =arrog=
=arrog=anci // =arroga=nci // =arroganc=i
=arrog=ant // =arroga=nt // =arrogant=
=arroj=
=arrojad=
=arrojadiz=
=arro=ll // =arroll=
=arro=p
=arrope=
=arrostr=
=arroy=
=arro=z
=arrozal=
=arrug=
=arrugad=
=arruin=
=arrull=
=arruma=c // =arrumac=
=arsenal=
=arsenic=
=art=e // =arte=
=artefa=ct // =artefac=t
=artej=
=art=er // =arte=r // =arter=
=art=eri // =arte=ri // =arter=i // =arteri=
=arte=rial // =arter=ial // =arteri=al // =arterial=
=artesan=
=artesanal=
=artesani=
=art=ic
=articul=
=articul=acion // =articula=cion // =articulacion=
=articul=ad // =articula=d
=articul=ar // =articula=r
=articul=ist // =articuli=st // =articulist=
=artific=e // =artifice=
=artifi=ci // =artific=i
=artifi=cial // =artific=ial // =artificia=l
=artific=ios
=artil=ler // =artiller=
=artiller=i
=artilu=gi // =artilug=i
=artimañ=

=art=ist // =artis=t // =artist=
=artis=tic // =artist=ic
=arzobis=p
=arzobis=pad // =arzobispa=d // =arzobispad=
=arzobispa=l
=as=
=asa=d
=asa=dor // =asador=
=asa=dur
=asalariad=
=asalt=
=asalta=nt // =asaltan=t
=asalta=nte // =asaltan=te // =asaltante=
=asam=ble
=asa=z // =asaz=
=asc=
=ascen=d
=ascendenc=i
=ascendent=
=ascendi=ente // =ascendient=e
=asce=ns // =ascen=s // =ascens=
=ascen=sion // =ascens=ion // =ascensio=n // =ascension=
=ascens=or // =ascenso=r
=asce=t // =ascet=
=asce=tic // =ascet=ic // =ascetic=
=asce=tism // =ascet=ism // =ascetis=m // =ascetism=
=ascu=
=ase=
=asead=
=asechan=z // =asechanz=
=asedi=
=asegur=
=asemej=
=asent=
=asentad=
=asentaderas=
=asentam=ient // =asentamie=nt
=asentim=ient // =asentimi=ent // =asentimient=
=asequibl=
=aser=cion // =aserci=on // =asercion=
=aser=r // =aserr=
=aserr=adur // =aserradu=r // =aserradur=
=ases=in // =asesin=
=asesin=at // =asesinat=
=ases=or // =asesor=
=ases=ori // =asesor=i // =asesori=
=ases=t
=asever=
=asfalt=
=asfaltad=
=asfixi=
=asfixian=t // =asfixiant=
=asi=atic // =asia=tic // =asiat=ic // =asiati=c
=asi=der // =asid=er // =aside=r
=asidu=
=asi=ent
=asign=
=asign=acion // =asigna=cion // =asignac=ion // =asignacion=
=asigna=tur // =asignatur=
=asi=l // =asil=
=asi=lad // =asil=ad // =asila=d
=asimetri=
=asimetric=
=asimil=
=asi=ri // =asir=i
=asi=st // =asis=t // =asist=
=asis=tenci // =asist=enci // =asisten=ci // =asistenci=

=asis=tent // =asist=ent // =asisten=t
=asist=ente // =asisten=te // =asistente=
=asm=
=asm=atic // =asmat=ic // =asmati=c // =asmatic=
=asn=
=asna=l
=asoci=
=asoci=acion // =asociaci=on // =asociacio=n // =asociacion=
=asoci=ad
=asol=
=asol=e
=asom=
=asombr=
=asombr=os // =asombro=s
=asona=d // =asonad=
=asona=nci // =asonan=ci // =asonanc=i
=asona=nt // =asonan=t
=asp=
=aspavi=ent
=aspec=t // =aspect=
=asp=er // =asper=
=asper=ez // =aspere=z // =asperez=
=aspi=r // =aspir=
=aspir=acion // =aspirac=ion // =aspiracion=
=aspi=rador // =aspir=ador // =aspirad=or // =aspirado=r // =aspirador=
=aspir=ant // =aspiran=t // =aspirant=
=aspir=ante // =aspiran=te // =aspirant=e
=aspir=in // =aspirin=
=asq=ue // =asqu=e
=asquero=s // =asqueros=
=ast=
=asterisc=
=asteroid=e // =asteroide=
=astill=
=astille=r // =astiller=
=astr=
=astrac=an // =astraca=n // =astracan=
=astral=
=astring=ent // =astringe=nt // =astringent=
=astring=ente // =astringe=nte // =astringent=e
=astrol=og // =astrolo=g // =astrolog=
=astrol=ogi // =astrolo=gi // =astrolog=i // =astrologi=
=astrol=ogic // =astrolo=gic // =astrolog=ic // =astrologi=c
=astronaut=
=astronauti=c // =astronautic=
=astronave=
=astronom=
=astronom=i
=astronom=ic
=astros=
=astuc=i
=astur=ian // =asturia=n // =asturian=
=astut=
=asu=m
=asunc=ion // =asuncion=
=asu=nt
=asu=st // =asust=
=at=
=atañ=
=atac=
=atac=ant // =ataca=nt
=atac=ante // =ataca=nte // =atacante=
=atad=
=atadu=r
=ataj=
=atalay=
=ataq=ue // =ataqu=e // =ataque=

=atardec=
=atardece=r
=ataread=
=atas=c // =atasc=
=atasca=der // =atascad=er // =atascade=r // =atascader=
=ataud=
=atav=i
=ate=
=ateis=m // =ateism=
=atemo=riz // =atemor=iz // =atemoriz=
=atemper=
=aten=
=atena=z // =atenaz=
=atenc=ion
=atend=
=atene=
=ateniens=
=ateniens=e
=atent=
=atent=ad // =atenta=d
=atenu=
=atenu=ant // =atenuan=t // =atenuant=
=ater=
=aterciope=lad // =aterciopel=ad // =aterciopela=d // =aterciopelad=
=ater=id // =ateri=d // =aterid=
=ater=r
=ater=rador // =aterra=dor // =aterrado=r // =aterrador=
=ater=riz // =aterri=z
=aterrizaje=
=aterroriz=
=ates=or // =ateso=r // =atesor=
=ates=t
=ates=tad // =atesta=d // =atestad=
=atestigu=
=atibor=r
=ati=c
=atild=
=ati=n // =atin=
=ati=nad // =atin=ad // =atinad=
=atis=b // =atisb=
=ati=z // =atiz=
=atlantic=
=atl=as
=atl=et // =atle=t
=atle=tic // =atletic=
=atle=tism // =atletis=m // =atletism=
=atmosf=er // =atmosfer=
=atmosfer=ic // =atmosferic=
=ato=ll // =atol=l // =atoll=
=atollad=er // =atollade=r // =atollader=
=atolondr=
=atolondra=d // =atolondrad=
=ato=m
=atomic=
=atomiz=
=ato=n
=atoni=t
=ato=nt
=ator=ment // =atorm=ent // =atormen=t // =atorment=
=atorn=ill // =atorni=ll
=atra=
=atrac=
=atracad=er
=atrac=ador // =atracad=or // =atracado=r
=atrac=cion // =atracci=on
=atrac=on // =atraco=n // =atracon=
=atrac=tiv // =atract=iv // =atracti=v

=atraga=nt
=atranc=
=atrap=
=atras=
=atrasad=
=atrav=es // =atraves=
=atraves=ad // =atravesa=d
=atray=ent // =atraye=nt
=atrev=
=atrevid=
=atrevim=ient // =atrevimi=ent // =atrevimie=nt
=atri=
=atribu=
=atribuc=ion // =atribucion=
=atribut=
=atril=
=atrinche=r
=atro=cidad // =atroci=dad // =atrocidad=
=atronad=or // =atronado=r // =atronador=
=atropell=
=atropell=ad // =atropellad=
=atro=z
=atuend=
=atu=n
=atur=d
=atur=did // =aturdid=
=auda=ci // =audac=i // =audaci=
=auda=z // =audaz=
=audi=bl // =audib=l
=audi=cion // =audic=ion // =audici=on
=audi=enci // =audien=ci // =audienc=i // =audienci=
=audiovisu=al // =audiovisua=l // =audiovisual=
=audi=t // =audit=
=audi=tiv // =audit=iv // =auditi=v
=audi=tor // =audit=or // =audito=r // =auditor=
=audi=tori // =audit=ori // =audito=ri // =auditor=i
=aug=e // =auge=
=aug=ur // =augur=
=augur=i // =auguri=
=august=
=aul=
=aul=l // =aull=
=aull=id
=aume=nt // =aumen=t
=aumenta=tiv // =aumentat=iv
=aun=
=aup=
=aur=
=aur=e
=aureo=l
=auricul=ar // =auricula=r
=aur=or // =auro=r // =auror=
=auscult=
=ause=nci // =ausen=ci // =ausenci=
=ause=nt // =ausen=t
=ause=nte // =ausen=te
=ausen=tism // =ausentis=m // =ausentism=
=auspi=ci // =auspic=i // =auspici=
=aust=er // =auster=
=auster=idad // =austerida=d // =austeridad=
=aust=ral // =austr=al // =austral=
=austral=ian // =australian=
=austriac=
=aut=
=autent=ic
=autentici=dad // =autenticidad=
=autentif=ic // =autentific=

=autent=iz // =autentiz=
=aut=ill // =autill=
=auto=
=autobio=grafi // =autobiog=rafi // =autobiogra=fi // =autobiografi=
=autobiog=rafic // =autobiogra=fic // =autobiografi=c // =autobiografic=
=autobom=b // =autobomb=
=autobus=
=autoca=r
=autocraci=
=autocr=at // =autocrat=
=autocr=atic // =autocrat=ic // =autocrati=c
=autocritic=
=autocto=n // =autocton=
=autodefens=
=autodetermin=acion // =autodetermina=cion // =autodeterminaci=on // =autodeterminacio=n // =autodeterminacion=
=autodidac=t // =autodidact=
=autodidac=tic // =autodidact=ic // =autodidacti=c
=autofinanci=acion // =autofinancia=cion // =autofinanciaci=on // =autofinanciacio=n // =autofinanciacion=
...