A.Z.R.A.E.L
A-Z Reconocedor Automático de EspañoL

Elena Álvarez Mellado

"Mas Yahveh descendió para ver la ciudad y la torre que los hombres estaban levantando y dijo: «He aquí que todos forman un solo pueblo y todos hablan una misma lengua, siendo este el principio de sus empresas. Nada les impedirá que lleven a cabo todo lo que se propongan. Pues bien, descendamos y allí mismo confundamos su lenguaje de modo que no se entiendan los unos con los otros». Así, Yahveh los dispersó de allí sobre toda la faz de la Tierra y cesaron en la construcción de la ciudad. Por ello se la llamó Babel, porque allí confundió Yahveh la lengua de todos los habitantes de la Tierra y los dispersó por toda la superficie."

Génesis 11

ÍNDICE

1.-MOTIVACIÓN Y ESTADO DE LA CUESTIÓN

1.1 Motivación

El programa AZRAEL nace al intentar responder a la pregunta de cómo un individuo es capaz de, dada una palabra, reconocer su idioma (o al menos aventurarlo), aun cuando no conozca su significado. No cabe duda de que ningún hispanohablante tendría problema en reconocer como español términos como *berenjena*, *troglodita*, *clarinete*, *pespunte* o *dinosaurio*. En este caso, los hablantes conocen el significado de las palabras, y por lo tanto saben con certeza que son palabras en español. Sin embargo, ¿son palabras en español términos como *médano*, *corrulla*, *zurumbo*, *sucinda*, *jareta* o *clánide*? Es probable que estas palabras fuesen desconocidas para muchos hispanohablantes, pero si les diésemos la definición de cada una no tendrían problema en admitirlas e incorporarlas como otra más. No obstante, si les advirtiésemos de que todas estas las palabras son españolas, salvo una de ellas que es inventada, y a continuación les pidiésemos que detectaran cuál es la farsante, tendrían problemas para detectar a la impostora, a menos que conociesen de antemano el significado de las demás. La mayoría se decantaría por una u otra palabra de forma más o menos aleatoria, o recurriría a un diccionario para identificar cuál de ellas no aparece recogida.

Supongamos ahora que entre esta última lista de palabras hubiesen aparecido otras palabras inventadas como *trock* o *whert*. En este caso no habría habido dudas acerca de cuál es la intrusa. Esto nos hace plantearnos mediante qué mecanismo reconoce un hablante el idioma de una palabra. ¿Por qué no tendríamos problemas en admitir *clánide*, pero rechazamos *trock* y *whert* como español? Puesto que las tres son palabras inventadas, no parece que detectemos el español simple y llanamente porque conozcamos previamente el significado asociado.

El siguiente texto extraído de *Rayuela* resulta iluminador en el tema que abordamos:

Apenas él le amalaba el noema, a ella se le agolpaba el clémiso y caían en hidromurias, en salvajes ambonios, en sustalos exasperantes. Cada vez que él procuraba relamar las incopelusas, se enredaba en un

grimado quejumbroso y tenía que envulsionarse de cara al nóvalo, sintiendo cómo poco a poco las arnillas se espejunaban, se iban apeltronando, reduplimiendo, hasta quedar tendido como el trimalciato de ergomanina al que se le han dejado caer unas fílulas de cariaconcia. Y sin embargo era apenas el principio, porque en un momento dado ella se tordulaba los hurgalios, consintiendo en que él aproximara suavemente su orfelunios. Apenas se entreplumaban, algo como un ulucordio los encrestoriaba, los extrayuxtaba y paramovía, de pronto era el clinón, las esterfurosa convulcante de las mátricas, la jadehollante embocapluvia del orgumio, los esproemios del merpasmo en una sobrehumítica agopausa. ¡Evohé! ¡Evohé! Volposados en la cresta del murelio, se sentía balparamar, perlinos y márulos. Temblaba el troc, se vencían las marioplumas, y todo se resolviraba en un profundo pínice, en niolamas de argutendidas gasas, en carinias casi crueles que los ordopenaban hasta el límite de las gunfias.

Este texto está escrito en glíglico, una lengua inventada por Cortázar. A pesar de que las palabras no tengan significado, el texto respeta la estructura morfológica del español, lo que hace que no resulte ininteligible. De hecho, si diésemos este texto a un estudiante extranjero de español de nivel medio, podría creer que las palabras inventadas existen realmente, ya que no hay nada en su apariencia que indique que no son españolas, como sí lo hay en *whert* o *trock*. Es sólo el hecho de conocer de antemano que nos encontramos antes un idioma inventado lo que nos hace saber que en realidad *noema*, *apeltronar* o *márulos* no tienen significado. Si bien es cierto que no podemos decir que este texto sea español estándar, tampoco podemos afirmar tajantemente que no lo sea: es en realidad una desviación del español convencional que respeta en su estructura las reglas de sintaxis y morfología. Nada impide que los hispanohablantes asignen al significante *noema* un significado concreto y pase a ser una palabra más del español, como lo son *dinosaurio* o *despunte*.

Por lo tanto, el significado no es determinante en la detección del idioma. Otro ejemplo de este hecho es que existen palabras en español que, teniendo un significado, son percibidas como ajenas a la lengua. Así, los extranjerismos como *whisky* son reconocidos como ajenos al español, a pesar de tener un significado asociado. De hecho, los extranjerismos que conservan su aspecto original producen vacilaciones a los hablantes en la escritura. Otros extranjerismos, en cambio, sufren un proceso de adaptación desde su forma en la lengua original a la de la de destino, y así, palabras extrajeras como *scanner* se modifican para adecuarlas a la forma que sí se admite como español, dando lugar a *escáner*. Por lo tanto,

existe un mecanismo que hace que un hispanohablante perciba como español *clánide*, *escáner* y *noema* pero no *whisky*, *scanner* y *trock*, al margen del significado que tengan.

Todo esto nos lleva a concluir que el mecanismo por el que detectamos una lengua se basa fundamentalmente en el aspecto de las palabras, lo que significa que es un mecanismo formal, independiente del significado. Partiendo de que la detección del idioma puede ser llevada a cabo desde un punto de vista exclusivamente formal, es posible crear un programa informático que detecte, a través de la forma de las palabras, si un texto está en español. El reto, por lo tanto, es doble: por un lado, conseguir llegar a un patrón que recoja cuál es la forma de las palabras en español (patrón que debe ser lo más general posible pero a la vez exhaustivo), y por otro, implementar ese patrón en un programa informático.

Por último, además de la necesidad práctica de reconocer un idioma, aparece el desafío de investigar por qué percibimos que una palabra pertenece a una lengua. Es decir, ¿las palabras en español tienen una lógica especial? Quizá las palabras en español deben empezar o acabar siguiendo una estructura determinada, o las sílabas admiten un patrón o un orden concreto y rechazan otro.

El español, a partir de diversos procesos de manipulación y mediante la sufijación y la prefijación, permite que el hablante cree palabras con un nuevo significado que el oyente es capaz de identificar, aunque sea la primera vez que las escuche. Esto puede hacer que en castellano exista un conjunto de posibilidades de inicio de palabra y otro conjunto de posibilidades de final de palabra, ya que si el principio o el final de la palabra no están dentro de ese conjunto, no es posible realizar ese proceso creativo. Por ejemplo en español existe un número muy limitado de palabras que no sean infinitivos que terminen en -ar, -er o –ir, y cuando aparece alguna como la citada *escáner,* nos causa una cierta confusión.

Así el español parece que necesita moldear las palabras para admitirlas en su seno, de forma semejante al proceso que sigue un canto rodado a la hora de incorporarlo a la lengua: por una parte, para dejar la palabra preparada para los procesos derivativos y compositivos necesarios, y, por otra, para que el hablante no tenga problemas al pronunciarla. Actualmente se incorporan muchas palabras del inglés, una lengua que no se pronuncia como se lee, lo que origina muchos problemas al hablante para saber cómo pronunciarla y cómo transcribirla.

Este trabajo es un primer estudio de la forma de la palabra en español, centrada sobre todo en la sílaba, que es la estructura que nos ha resultado más relevante para encontrar la lógica de las palabras en español. A partir de este estudio quizá pudiéramos establecer algún procedimiento que determinase, dada una palabra extranjera que se quiere incorporar a la lengua, la forma que debería tener en español, ahorrándonos un periodo transitorio más o menos largo en el que convive la versión inicial y diversas transformaciones hasta llegar a la forma final.

En el fondo la motivación principal al realizar este trabajo es reflexionar sobre la siguiente cuestión: ¿qué hace que una palabra sea española?

1.2 Estado de la cuestión

El interés por la naturaleza del lenguaje es casi tan antiguo como el lenguaje mismo. La primera gramática de la que se tiene testimonio es la gramática del sánscrito de Panini, que se remonta al siglo V a.C. aproximadamente. En occidente, es en Grecia donde nacen los primeros estudios gramaticales y donde aparece por primera la diatriba sobre la relación entre significante y significado, que retomaría Saussure a principios del siglo XX.

Pero más allá de cuestiones filosóficas sobre la naturaleza del lenguaje, el problema de las diferencias entre lenguas ha surgido de la propia convivencia e interacción entre poblaciones lingüísticamente distintas. Los orígenes de la comparación lingüística se remontan a la Edad Media. Uno de estos primeros trabajos comparativos lo realizó Dante Alighieri, clasificando las lenguas en base a la forma de la palabra *sí*, distinguiendo las que procedían de la forma latina *sic*, de las demás. A finales del siglo XVI, en Diatriba de Europaeorum Linguis, Joseph Justus Scaliger hace una nueva clasificación de las lenguas, esta vez caracterizándolas por la palabra "dios". Estos primeros trabajos de comparación de lenguas eran, por tanto, clasificaciones léxicas y constituyen el precedente de la Tipología Lingüística.

A lo largo del siglo XVII aparecen los primeros intentos de superar las limitaciones que suponían las diferencias lingüísticas mediante artefactos mecánicos. Este hecho se debe a dos causas: por un lado, la desaparición progresiva del latín como lengua universal para la

comunicación científica, y por otro, el interés de racionalistas como Leibniz y Descartes en crear un lenguaje artificial, lógico y universal que estuviese libre de la ambigüedad de los lenguajes naturales. La idea era crear lenguajes con códigos numéricos en vez de palabras, para poder crear textos universales que cualquiera pudiera leer disponiendo de un diccionario en el que cada código remitiese al término equivalente.

Las propuestas de posibles lenguajes artificiales numéricos se sucedieron hasta el siglo XX, y algunos de estos diccionarios mecánicos se llevaron a cabo durante 1920 y 1930. Sin embargo, no es hasta los años cuarenta cuando se desarrollaron las primeras máquinas para el desciframiento de código, traducción y detección automática del lenguaje. El matemático inglés Alan Turing (que había trabajado durante la Segunda Guerra Mundial en el desciframiento de los mensajes secretos alemanes) defendía que si un ordenador podía descifrar un código, también podría traducir un idioma extranjero. Tras esta afirmación se encuentra la noción de que un texto escrito en un idioma extranjero no es más un mensaje en un código desconocido pero susceptible de ser descifrado. La misma aproximación se encontraba en las cartas del matemático americano Warren Weaver del año 1947:

(…) Uno se pregunta si el problema de la traducción podría ser tratado como un problema de criptografía. Cuando miro un artículo en ruso, me digo: "Esto es en realidad inglés, pero está codificado en símbolos extraños. Ahora debo descodificarlos." (…)

El desarrollo de la informática y las propuestas de la Gramática Generativa de los años posteriores supusieron una revolución en la Lingüística. Los ordenadores ofrecían un filón de posibilidades en el tratamiento del texto, la traducción, y la extracción de información, mientras que Internet, a su vez, suponía una fuente inagotable de textos y material lingüístico. La Lingüística Computacional y el Procesamiento de Lenguaje Natural (PLN) nacieron como ramas mixtas en las que confluían el estudio lingüístico y el uso de herramientas informáticas. A partir de estas nuevas aproximaciones se han ido desarrollando diversos modelos de traducción y detección de idioma, si bien algunos problemas como la polisemia y la desambiguación no han podido ser resueltos todavía, y los resultados de los traductores automáticos no han logrado acercarse a la labor de traducción humana.

Actualmente existen múltiples detectores y traductores automáticos a libre disposición en Internet, siguiendo distintas aproximaciones para la detección automática del idioma, pero los detectores manejados antes de la creación de AZRAEL funcionan en base a uno de los siguientes cinco métodos:

A. Diccionario

B. Conectores

C. Alfabeto

D. Frecuencia de aparición de letras

E. Modelo de n-grama

A. Diccionarios

La opción más intuitiva para detectar un idioma sería partir de un inventario de las palabras existentes en esa lengua (un lexicón) y comprobar una a una si las palabras que aparecen en el texto a detectar están incluidas en el lexicón. Si aparecen en el lexicón todas las palabras (o un porcentaje lo suficientemente alto para que sea concluyente) sabremos en qué lengua está el texto.

Esta aproximación tiene el inconveniente del manejo de un inventario de palabras del castellano, que, por muy vasto que fuese, siempre resultaría incompleto y habría de estar en constante revisión para incorporar términos nuevos, lo que complica la detección. Además, bajo este método subyace la idea de que un texto está en una lengua sólo si las palabras que lo componen pertenecen al conjunto de palabras de esa lengua. Esto presupone que es sólo la existencia de un significado asociado lo que hace que una palabra pueda pertenecer a una lengua. Sin embargo, como hemos visto en el apartado anterior, la propia forma de la palabra ya impone una restricción para la formación de palabras en una lengua: no hay nada que impida que *clánide* pase a ser una palabra en español, mientras que es improbable que *whert* o *trock* sean admitidas o acuñadas por hispanoparlantes.

B. Conectores

Un segundo método consiste en la detección de conectores (preposiciones, determinantes y conjunciones). Los conectores son palabras breves, invariables,

reacias a cambiar con el paso del tiempo, indispensables en un texto y constituyen un conjunto cerrado, así que una posible aproximación consistiría en caracterizar una lengua mediante su conjunto de conectores. Puesto que los conectores no suelen superar las cinco letras, no habría más que, dado un texto cualquiera, seleccionar las palabras con menos de cinco letras y comprobar si pertenecen a este conjunto. Si un porcentaje significativo de las palabras de menos de cinco letras del texto pertenece a la lista de conectores, podremos afirmar que el texto está en ese idioma.

Este método es aplicable a todas las lenguas (todas tienen conectores), es más sencillo y resulta menos costoso en recursos que el anterior, ya que pasamos de depender de un inventario de todas las palabras de una lengua (que por su propia naturaleza siempre será incompleto) a manejar unas pocas decenas de palabras. No obstante, este sistema tiene una carencia fundamental: al tener en cuenta sólo un subconjunto muy restringido de las palabras de una lengua, sólo es válido para textos, nunca para palabras sueltas. Sin embargo, como ya se apuntó en el apartado anterior, un hablante es capaz de reconocer un idioma sólo por el aspecto de las palabras, aunque aparezcan aisladas. La detección mediante conectores no nos dice por qué *clánide*, en caso de ser acuñada, podría ser una palabra perfectamente aceptable en castellano que ningún nativo tendría problema en admitir, mientras que *whert* y *trock* no.

C. Alfabeto

Quizá la forma más simple de hacer un reconocedor de idioma (aunque curiosamente de las menos extendidas) sea mediante los caracteres que son exclusivos de una lengua. Así, por ejemplo, el armenio, el georgiano y el tailandés tienen alfabetos únicos, que no comparten con otras lenguas, por lo que sería fácil identificarlos siguiendo este criterio. Sin embargo, este recurso no está limitado sólo a las lenguas con alfabetos únicos, ya que la gran mayoría de lenguas que usan un alfabeto más extendido (por ejemplo, el latino) también cuentan con letras, acentos o tildes propios que no aparecen en otras lenguas y que por lo tanto permitirían este método de detección. De este modo, sabríamos que un texto es español si contiene la letra Ñ. El problema de este sistema no es sólo que deje de lado criterios

morfológicos, sino que además dependemos de que aparezcan en el texto a detectar las letras que caracterizan a esa lengua.

D. Frecuencia de aparición de letras

Una cuarta forma posible de caracterizar una lengua es por la frecuencia de aparición de letras. Así, detectaríamos el idioma de un texto en base al número de veces que aparece una o varias letras concretas que sean particularmente frecuentes en esa lengua; este método puede extenderse a pares de letras que suelan aparecer juntas. Se trata por lo tanto, de una aproximación estadística, ya que tiene detecta a partir de los porcentajes de aparición de determinadas letras. La desventaja reside en que sólo es válido para textos, no para palabras sueltas. Por otro lado, tiene el inconveniente de que, según ante qué tipo de textos nos encontremos, pueden aparecer más o menos veces determinadas letras. Un texto escrito en futuro en el que abunde la perífrasis IR + INFINITIVO tendrá, debido a la presencia de infinitivos, un número mucho más alto de R que un texto escrito en pasado.

E. Modelo de n-grama

Por último, la detección del idioma se puede realizar siguiendo un modelo de n-gramas. Este modelo permite, dada una secuencia de caracteres, predecir cuál es el siguiente carácter que aparecerá con más probabilidad. A partir de un corpus de textos lo suficientemente amplio, es posible llegar a patrones de distribución de probabilidad que caractericen a esa lengua, de tal manera que permitan la detección de un idioma. Un modelo así permitiría saber que, dada la cadena de caracteres "cadena de caract", la letra más probable de aparición en español es una E. Se trata, por tanto, de un análisis probabilístico de los elementos que aparecen en una secuencia de caracteres en un idioma determinado.

AZRAEL comparte algunos rasgos con estos métodos, si bien se trata de una aproximación distinta y novedosa al problema de la detección automática del lenguaje. Esta nueva aproximación supone algunas mejoras respecto a otros enfoques, y al mismo tiempo plantea nuevos retos a resolver. En cualquier caso, ninguno de estos planteamientos es totalmente satisfactorio por sí solo, ya que todos resuelven algunos aspectos, pero acarrean problemas en otros. Quizá el mejor enfoque para la detección de idioma sea desde una

perspectiva sinérgica, en el que varios mecanismos analicen distintos aspectos del texto, aumentando la fiabilidad de la detección.

2.-OBJETIVOS DE LA INVESTIGACIÓN

La creación de AZRAEL responde a la necesidad actual de desarrollar instrumentos informáticos que nos ayuden en el tratamiento automático del lenguaje. La detección del idioma es fundamental en el uso de herramientas tan extendidas como los procesadores de texto y los correctores automáticos. Asimismo, la Red pone a nuestra disposición una inmensa cantidad de información en múltiples idiomas que necesita ser procesada lingüísticamente para poder sacarle partido.

Nuestro propósito es, por lo tanto, crear un detector de idioma eficaz que suponga una mejora respecto a los ya existentes. Esto quiere decir que la aproximación con la que nos acercaremos al tema será distinta de las que se han planteado hasta ahora, ya que nos centraremos fundamentalmente en la detección en base a la estructura silábica, lo que supone una novedad a este campo de investigación.

El aspecto básico sobre el que se pretende realizar la detección del idioma es la forma de las palabras, ya que como se dijo anteriormente (v. MOTIVACIÓN Y ESTADO DE LA CUESTIÓN), es este (y no la existencia de significado) el mecanismo fundamental por el que un hablante reconoce el idioma de una palabra. La aspiración, por lo tanto, no es que detecte exclusivamente lo que un diccionario o la propia RAE considerarían como español, sino que nuestro objetivo es mucho más ambicioso, ya que queremos que AZRAEL admita como español todo aquello que un hablante admitiría, es decir, todo aquello que tenga apariencia de español, al margen de que tenga significado o no. Un detector de estas características aceptaría lo que normativamente se considera español, pero también otras producciones menos convencionales, como neologismos, términos propios de la jerga callejera, palabras sin sentido propias de juegos y canciones infantiles (como *miliquituli* o *lerê*), palabras inventadas (como las del texto en glíglico que mencionábamos al comienzo), etc.

En ningún caso este enfoque es abusivo: no pretendemos hacer un programa que distinga español canónico del que no lo es, por lo tanto, no podemos restringir un idioma a lo que normativamente se considera correcto cuando el objetivo es la creación de una herramienta

práctica que detecte desde textos literarios o académicos hasta producciones propias del lenguaje coloquial.

El segundo objetivo que perseguimos es la realización de un estudio en profundidad de la estructura de las palabras del español en base a la sílaba. Este estudio nos proporcionará la clave fundamental para la detección del español. La estructura silábica del español apenas sí ha sido abordada previamente con la profundidad que el tema merece, aun cuando resulta una fuente inagotable de información para caracterizar el idioma y para extraer datos sobre la naturaleza de la palabra. Por tanto, con el estudio de la sílaba no sólo buscamos llegar a un patrón que se pueda implementar en un programa informático, sino también comprobar cuánta información podemos sacar de la palabra a partir de su estructura silábica.

3.-DATOS

Para la realización de AZRAEL ha sido necesario hacer un estudio previo de la estructura silábica del español, ya que la sílaba es el elemento fundamental de detección de nuestro programa. Como no se han encontrado descripciones completas de la sílaba que pudieran ser útiles para nuestro objetivo, hemos realizado nuestro propio estudio exhaustivo del tema, partiendo de un corpus del español de más de 600.000 palabras. A partir de este corpus, hemos analizado qué combinaciones de letras son posibles dentro de las sílabas del español, y qué combinaciones están prohibidas. Además de mantener la división tradicional entre consonantes y vocales, hemos llegado a una clasificación de las consonantes según las posibles funciones y posiciones que admiten.

La combinación que todas las consonantes admiten es la estructura CONSONANTE+VOCAL, salvo la letra Q, que como veremos más adelante, es especial.

Algunas consonantes admiten que se intercale entre ellas y la vocal otra consonante, que hemos llamado *consonantes intercaladas*. Estas consonantes son la R y la L. Aquellas consonantes que admiten que las acompañe indistintamente una u otra, las hemos llamado *consonantes tipo 1*, y son la B, la C, la F, la G y la P. Las *consonantes tipo 2* son aquellas que sólo admiten como compañera a la R, y son la T y la D. No es casual que sean justo estas dos letras las que pueden intercalarse: los sonidos a los que ambas letras remiten son sonidos líquidos, que están a medio camino entre los sonidos consonánticos más puros (oclusivos y fricativos) y los vocálicos. Son sonidos que presentan características espectrográficas semejantes a las de una vocal, y que pueden constituir por sí solas núcleo silábico en otros idiomas. De hecho, en gramáticas de algunas lenguas como el sánscrito la L y la R son consideradas vocales. Por lo tanto, existe una explicación fonética para que sean estas dos letras y no otras las que pueden formar combinaciones con otras consonantes.

Las vocales pueden encabezar solas la sílaba como en *o-ro*, pero si van precedidas de una consonante o grupo consonántico se unen a ellas, como en *to-ro* y *clo-ro*. Si una *vocal fuerte* (A, E, O, Í, Ú) va seguida de una *vocal débil* (I,U) ambas tienden a unirse, formando parte de la misma sílaba (diptongo descendente). En el caso inverso (vocal débil seguida de fuerte,

diptongo ascendente) o si ambas vocales son débiles también permanecen unidas. Sin embargo, dos vocales fuertes juntas (hiato) forman parte de sílabas distintas.

Ejemplos:

pei-nar	*drui-da*
hia-to	*a-é-re-o*

Si bien todas las consonantes pueden ser principio de sílaba, no todas pueden ser final de sílaba. Sólo las siguientes consonantes pueden desempeñar esta función sin restricciones: R, S, L, N, D, Z. Son las *consonantes finales*, que pueden no sólo ser final de sílaba sino también final de palabra. Es necesario destacar, no obstante, que las consonantes finales sólo son final de sílaba si no van seguidas de vocal. En este caso, la consonante final abandonaría la sílaba precedente para unirse a la vocal de la sílaba siguiente y convertirse en la consonante que encabeza la sílaba.

Ejemplo:

car - tón frente a *ca - re - ta*

Las vocales no tienen restricciones para ser final de sílaba.

Analizando las palabras del corpus descubrimos que existen algunas consonantes que pueden comportarse como consonantes finales, pero sólo en determinados contextos, es decir, si van seguidas de una consonante en concreto. Presentamos a continuación una tabla que recoge las consonantes que pueden ser ocasionalmente final de sílaba. Se incluyen en esta tabla la consonante que aparece en posición final de sílaba, el contexto en que se da (es decir, qué consonante se exige que siga), el número de palabras del corpus manejado en que se da esta combinación y ejemplos.

CONSONANTE FINAL DE LA SÍLABA	CONTEXTO CONSONÁNTICO	NÚMERO DE CASOS RECOGIDOS EN EL CORPUS	EJEMPLOS
C	t	8627	ac-tuar
	c	2189	ac-ce-der
	n	275	a-rác-ni-do
	m	28	drac-ma
	s	19	fuc-sia
F	t	332	dif-te-ria
	g	18	af-ga-no
	n	8	haf-nio
G	n	2487	ig-no-rar
	m	696	dia-frag-ma
	d	42	a-míg-da-las
M	p	21054	pom-pa
	b	16549	bom-ba
	n	564	a-lum-no
	m	52	gam-ma
	l	10	drum-lin
	h	6	brom-hí-dri-co
P	t	2972	a-cep-tar
	c	413	op-ción
	n	121	hip-no-sis
T	m	194	at-mós-fe-ra
	l	115	at-le-ta
	n	72	et-nia

CONSONANTE FINAL DE LA SÍLABA	CONTEXTO CONSONÁNTICO	NÚMERO DE CASOS RECOGIDOS EN EL CORPUS	EJEMPLOS
B	s	1024	ob-ser-var
	t	413	sub-te-rráneo
	v	367	ob-vio
	d	327	ab-di-car
	j	269	ob-je-to
	c	250	ob-ce-car
	y	175	ab-yec-to
	n	89	ob-nu-bi-lar
	m	66	sub-ma-ri-no
X	p	2569	ex-pan-dir
	t	2459	ex-tra-er
	c	1573	ex-cep-ción
	h	397	ex-ha-lar
	q	53	ex-qui-si-to

En las combinaciones de la tabla podemos distinguir tres grupos:

a. En color rosa, las combinaciones que se dan en español con frecuencia y que son percibidas como perfectamente admisibles por los hablantes, como *a-cep-tar*, *a-lum-no* o *pom-pa*. A estas consonantes que se comportan como consonantes finales en contextos concretos las llamaremos *consonantes pseudofinales*. En ningún caso pueden estas consonantes pseudofinales ser final de palabra, sólo de sílaba si se dan las condiciones adecuadas.

b. En azul, las combinaciones mucho menos frecuentes y que son percibidas como ajenas al idioma por los hablantes, como *drac-ma*, *brom-*

hí-dri-co, *a-rác-ni-do* o *dif-te-ria*. Aparecen sólo en extranjerismos o en palabras que se han incorporado tardíamente al español, como los cultismos provenientes del griego que se recuperaron para formar términos del ámbito científico.

c. Existe un tercer grupo de consonantes (en verde) que pueden ser final de sílaba (aunque nunca de palabra) y que parecen admitir casi cualquier letra detrás, formando combinaciones poco frecuentes pero siendo percibidas como español perfectamente admisible. Es el caso de consonantes como la B, que admite ser final de sílaba cuando va seguida, por ejemplo, de J (como en *ab-ju-rar*), de N (*ob-nu-bi-lar*) o de Y (*sub-ya-cer*). También la X es especialmente flexible y es final de sílaba en contextos como Q (en *ex-qui-si-to)*, T (en *ex-tra-er*) o H (*ex-ha-lar*).

¿Cómo se explica que estas combinaciones aparentemente anómalas resulten frecuentes y totalmente aceptables? El motivo es que en este tercer grupo, los fenómenos morfológicos de derivación se superponen a las reglas de estructuración silábica. En todas las combinaciones en las que la X es final de sílaba, aparece formando parte del prefijo *ex-*. Como la partícula *ex-* es un prefijo y aporta significado por sí mismo, los hablantes admiten que pueda ir seguida por cualquier consonante. Es decir, aunque *ex-* sea una sílaba, las reglas silábicas no se aplican en este caso, ya que *ex-*, sin ser independiente, es una partícula con significado autónomo. Esto quiere decir que *exhalar* no es una palabra válida en castellano porque la combinación X+H sea válida, sino porque lo que es válido es que *ex-* vaya seguido de cualquier combinación.

El mismo fenómeno que se da en *ex-* se produce también en *sub-*. Sin embargo, la B como consonante final de sílaba no sólo nos la encontramos cuando forma parte del prefijo *sub-*. Todos los demás casos en los que aparece la B como consonante final son como parte de las sílabas *ob-* y *ab-*. En español, estas sílabas ya no son percibidas por los hablantes como prefijos con significado autónomo., sin embargo, ambas partículas eran prefijos en latín. Esto quiere decir que para los

hablantes de latín, *ab-* y *ob-* sí tenían un significado propio. Por lo tanto, de igual manera que los hispanohablantes no tienen problema en admitir cualquier letra detrás de los prefijos *ex-* y *sub-*, los latinos podían construir palabras en las que *ab-* y *ob-* fueran seguidos de cualquier combinación de letras. De hecho, los prefijos *ex-* y *sub-* también nos vienen del latín, lo que significa que tenemos algunas palabras como *subjuntivo*, en las que sub- ha dejado de entenderse como prefijo con significado autónomo, ya que han sido heredadas tal cual del latín (mientras que *submarino* sí es entendido como "algo bajo el mar", *subjuntivo* no se percibe como "algo por debajo de la juntura", aunque ese fuera su origen etimológico).

Aunque las palabras con *ab-* y *ob-* como prefijos son abundantes, todas provienen del latín, ninguna es de creación posterior. Esto quiere decir que en el momento en que los hablantes perdieron la percepción de ab- y ob- como prefijos, dejaron de crearse palabras nuevas con esta forma, lo que supone que la posición de B como final de sílaba era percibida como extraña. Las palabras que ya existían se conservaron y han llegado hasta nosotros como fósiles que muestran lo que en otro tiempo sí había sido percibido como admisible, pero esa estructura, una vez que dejó de hablarse el latín, no caló en el patrón silábico del español. Este hecho aporta una gran cantidad de información etimológica, ya que podemos afirmar que todas las palabras que tengan el prefijo *ab-* y *ob-* vendrán del latín.

Sólo se han considerado las combinaciones del grupo I como combinaciones silábicas válidas en español. Las combinaciones de los grupos II y III se han tenido en cuenta a la hora de escribir el código del programa (ver METODOLOGÍA) y reciben un tratamiento especial. Por un lado, a partir de las combinaciones del grupo II podemos extraer una inmensa cantidad de información etimológica de las palabras (idioma de procedencia, épocas de incorporación al castellano, etc). Por ejemplo, todas las palabras que contienen el grupo *tl* provienen necesariamente del griego (como en *atleta*) o del náhuatl (como en *tlacoyo*). De forma aproximada (puesto que se sale de los objetivos de la investigación) podemos proponer la siguiente lista de grupos consonánticos que no son propios del

español pero que aparecen en palabras españolas, así como el origen etimológico de las palabras que presentan esos grupos.

TL: Griego o náhuatl

Ejemplo: at-le-ta, tla-co-yo

TM: Griego

Ejemplo: at-mós-fe-ra

TN: Griego

Ejemplo: et-nia

Pn: Griego

Ejemplo: hipnosis

Ft: Griego o árabe

Ejemplo: dif-te-ria, muf-tí

Gm: griego

Ejemplo: dia-frag-ma

Gd: Griego

Ejemplo: a-míg-da-las

Cn: Griego

Ejemplo: a-rác-ni-do

It: Latinismo

Ejemplo: dé-fi-cit

Por otro lado, de las combinaciones del grupo III obtenemos información morfológica de la naturaleza de los prefijos en castellano. Todo esto se ha tenido en cuenta a la hora de realizar el programa.

Finalmente, nos encontramos con algunas letras que tienen un patrón de comportamiento particular:

- La S tiene la peculiaridad de que puede ir cerrando una sílaba detrás de una consonante final o a un prefijo, como en *trans-por-te*, *abs-trac-to*, *sols-ti-cio* o *pers-pi-caz*. La hemos llamado *consonante final total*, ya que es la única consonante que tiene la capacidad de unirse a una consonante final para acabar la sílaba. La mayoría son términos heredados del latín (aunque no todos, como en *sáns-cri-to*), y de hecho muchos tienden a simplificar el grupo consonántico perdiendo la consonante final y conservando sólo la S. Así, por ejemplo, la forma *obs-cu-ro* ha ido perdiéndose a favor de *os-cu-ro*. Esto nos demuestra que los hablantes perciben como anómalas estas combinaciones consonánticas y tienden a simplificarlas para adaptarlas a la estructura silábica habitual del español. Como en el caso de los prefijos, las palabras que nos llegan con esta estructura son fósiles que nos aportan información sobre su origen y nos revelan cómo era la estructura silábica en otras lenguas.

- La Y la hemos considerado consonante y vocal, porque puede desempeñar un papel u otro según el contexto silábico en el que se encuentre.

- La Q sólo puede funcionar si va seguida de una U, lo que quiere decir que por sí sola no es una letra autónoma e independiente.

- Por último, la LL, la RR y la CH son combinaciones de dos letras que representan un único sonido, pero que a diferencia de la Q, pueden funcionar también solas. Son las *consonantes emparejadas*.

Este patrón de la sílaba en español que acabamos de describir ha sido la base para la creación del silabeador. En el apartado de METODOLOGÍA explicaremos cómo el código del programa recoge toda la información teórica que hemos expuesto.

4.- METODOLOGÍA

AZRAEL propone un método novedoso en el campo de la detección: la sílaba como forma de reconocer un idioma. Por lo tanto, el programa necesitará de la creación de un programa anexo de silabeo, en el que implementaremos el modelo teórico de la sílaba que se detalló anteriormente (ver DATOS). El objetivo es que, si el silabeador es capaz de silabear un texto dado, entonces podremos afirmar que nos encontramos ante un texto en español. Si el silabeador no puede silabear el texto, significará que no sigue la estructura silábica del español, y por lo tanto AZRAEL no lo detectará como español.

La detección del idioma a través de la sílaba conlleva múltiples ventajas. Por un lado, no dependemos de un inventario del léxico de una lengua, que resultaría muy costoso de manejar desde el punto de vista informático y que siempre resultaría incompleto. Por otro lado, el léxico de una lengua está en constante cambio, aparecen palabras nuevas, otras caen en desuso, pero la estructura de la sílaba se mantiene estable durante siglos. No nos es útil un diccionario del siglo XIX para leer un texto actual, pero las normas silábicas que regían el español en el siglo XIX siguen siendo válidas en el español de hoy. Además, la detección del español a través de la sílaba nos permite acercarnos a textos menos convencionales, ya que las reglas silábicas que rigen el español normativo y académico rigen también la estructura de los lenguajes inventados, la jerga callejera o las palabras propias de los juegos de niños. Por último, el estudio de la sílaba nos permitirá sacar gran información de la naturaleza de las palabras, de su etimología y de los procesos morfológicos que ocurren en los préstamos léxicos desde una lengua a otra.

Puesto que AZRAEL es un detector de texto escrito, en la realización del programa se ha tomado el carácter tipográfico como elemento mínimo de estudio. Los caracteres se combinan siguiendo un patrón determinado, conformando una sílaba, que es la unidad en base a la cual se analiza el texto. Es decir, el carácter es la unidad estructural, mientras que la sílaba es la unidad funcional. Metafóricamente, podría verse como los átomos de un ser vivo: el conjunto de tipos de átomos que conforman un ser vivo es cerrado. Existe una inmensa cantidad de posibles moléculas que esos átomos podrían formar combinándose unos con otros, pero lo cierto es que sólo encontramos unas moléculas determinadas en el

interior de los seres vivos; sólo son válidas algunas de las innumerables combinaciones que serían posibles. De igual modo, los caracteres que existen en una lengua podrían combinarse de múltiples formas distintas, pero la realidad es que sólo encontramos un subconjunto de todas las combinaciones que teóricamente serían posibles.

Por otro lado, hemos definido palabra como la secuencia de caracteres entre dos espacios en blanco. Esto significa que para nosotros una secuencia como *dáselo* es una sola palabra y *se lo da* son tres. Este enfoque deja al margen otras aproximaciones semánticas o morfológicas sobre la problemática del tema, pero resulta muy eficaz desde el punto de vista del procesamiento automático del texto, ya que es sencillo, útil y sobre todo completo.

El método seguido para la creación de AZRAEL ha consistido en, una vez realizado el estudio teórico descrito en DATOS, pasar el conocimiento de la estructura silábica del español a lenguaje de programación Prolog para lograr un programa que discrimine el español en base a la estructura de la sílaba.

La estructura del programa es la siguiente: metemos en AZRAEL un texto entre comillas (las comillas son necesarias para que Prolog lea el texto en código ANSII). Ese texto es procesado para aislar cada palabra en una lista con los caracteres ANSII correspondientes. A continuación se eliminan los signos de puntuación y se manda cada palabra al Silabeador.

El Silabeador es un programa auxiliar de AZRAEL que divide la palabra en sílabas. El Silabeador recibe la palabra en forma de lista de caracteres sueltos, es decir, recibe la palabra deletreada, y analiza letra a letra las combinaciones que forman las letras para comprobar si se corresponden con la estructura silábica del castellano. Si la respuesta es afirmativa, pasa a la siguiente palabra. Si la estructura silábica no se corresponde con la estructura del español, entonces el Silabeador no puede silabear la palabra y se la devuelve a AZRAEL.

Una vez que este procedimiento lo ha llevado a cabo con todo el texto, existen dos posibles caminos, según hayamos decidido pedir más o menos información a AZRAEL. Una opción es que AZRAEL nos devuelva sin más la lista de las palabras que no ha podido reconocer (las que el Silabeador no ha podido silabear). La otra opción es que la lista de palabras sin

reconocer la mande a un segundo programa auxiliar, ESDRA (Etimólogo Selectivo Del Reconocedor Automático).

La labor de ESDRA consiste en intentar averiguar si, a pesar de no responder a la estructura silábica del español, las palabras que el Silabeador ha descartado pueden ser préstamos de otras lenguas. ESDRA buscará si las palabras no reconocidas contienen alguna combinación consonántica propia de algún idioma de la que haya podido ser importada. Si la estructura de la palabra desconocida no se corresponde con ninguna de las posibilidades de ESDRA, le asignará un origen desconocido. Una vez que ESDRA ya ha comprobado todas las palabras, devuelve a AZRAEL la lista de las palabras descartadas con su posible origen asociado. Lo que finalmente nos devolverá AZRAEL serán dos listas: una con las palabras no reconocidas por el Silabeador, y la segunda con los orígenes etimológicos posibles que ESDRA les ha asignado a los préstamos (las palabras que no sean préstamos aparecerán clasificadas como de origen desconocido).

Expondremos a continuación el código completo del programa, comentando los aspectos más relevantes de su funcionamiento:

AZRAEL: Código fuente del programa

1.- Programa Principal

% El texto debe ir entre comillas para que el programa lo procese como una lista de códigos ANSII. El predicado español/1 dice que algo es español si hacemos el procedimiento lista/2 y el comprobar_palabras/1.

- Función español(Texto,Restos,Origen).
  - Texto: Texto a analizar,
  - Restos: Lista de palabras no españolas
  - Origen: Lista de palabras con su posible origen etimológico

```
español(X,Restos,Origen):-
        español(X,Restos),
        origen_etimológico(Restos,Origen).
```

- Función español(Texto,Restos).
  - Texto: Texto a analizar,
  - Restos: Lista de palabras no españolas

```
español(X,Restos):-
    transcribir(X,W),
    !,
    lista(W,Y),
    comprobar_palabras(Y,N),
    recuperar_descartes(N,Restos).
```

- Función origen_etimológico(Palabras,Origen).
  - Palabras: Lista de palabras a estudiar
  - Origen: Lista de palabras con su posible origen etimológico

```
origen_etimológico([],[]).

origen_etimológico([Restos|MasRestos],[Origen|MasOrigen]):-
                    atom_chars(Restos,Deletreo),
                    combinaciones(Restos,Deletreo,Origen),
                    !,
                    origen_etimológico(MasRestos,MasOrigen).
```

2.- Funciones

- Función combinaciones(Restos,Letras,Origen).
  - Restos: Palabras no reconocidas a estudiar

- Letras: Letras identificadoras de origen etimológico
- Origen: Origen etimológico correspondiente a dicha combinación.

%Estas son las combinaciones que ESDRA es capaz de reconocer en los préstamos y el idioma al que las asigna

```
combinaciones(Restos,[J],desconocido(Restos)).

combinaciones(Restos,[t,l|W],griego_o_náhuatl(Restos)).

combinaciones(Restos,[t,m|W],griego(Restos)).

combinaciones(Restos,[t,n|W],griego(Restos)).

combinaciones(Restos,[p,n|W],griego(Restos)).

combinaciones(Restos,[f,t|W],griego_o_árabe(Restos)).

combinaciones(Restos,[g,m|W],griego(Restos)).

combinaciones(Restos,[g,d|W],griego(Restos)).

combinaciones(Restos,[c,n|W],griego(Restos)).

combinaciones(Restos,[i,t],latinismo(Restos)).

combinaciones(Restos,[X,Y|W],Origen):-
                    combinaciones(Restos,[Y|W],Origen).
```

- Función recuperar_descartes(Deletreada,Palabra).
  - Deletreada: Palabras no reconocidas deletreadas,
  - Palabra: Palabras no reconocidas

```
recuperar_descartes([],[]).

recuperar_descartes([N|M],[Restos|MasRestos]):-
        atom_codes(Restos,N),
        recuperar_descartes(M,MasRestos).
```

- Función transcribir(Entrada,Salida).
  - Entrada: Texto a analizar,
  - Salida: Texto convertido a minúsculas

% transcribir/2 convierte el texto a minúsculas para que pueda ser procesado por el Silabeador

```
transcribir([],[]).

transcribir([W|Y],[W|Z]):-
        W<65,
        transcribir(Y,Z).

transcribir([W|Y],[A|Z]):-
        W=<90,
        W>=65,
        A is W + 32,
        transcribir(Y,Z).

transcribir([W|Y],[A|Z]):-
        W=<220,
        W>=193,
        A is W + 32,
        transcribir(Y,Z).


transcribir([W|Y],[W|Z]):-
        W>90,
        transcribir(Y,Z).
```

3.- Métodos Auxiliares

% El procedimiento lista/2 elimina espacios y signos de puntuación. Devuelve una lista que contiene las listas individuales de cada palabra con sus caracteres ANSII, pero ya sin espacios ni signos de puntuación.

```
lista(X, Y):- corte(X, Y, _, []), !.

%Reglas base
corte([], Resprov, Resfinal, []):-append(Resprov, [], Resfinal).
corte([], Resfinal, Resprov, Sublista):-Sublista\=[], append(Resprov, [Sublista],
Resfinal).

% Reglas recursivas
corte([SignPunt|Resto], Resfinal, Resprov, Sublista):-
                        signPunt(SignPunt),
                        append(Resprov, [Sublista], Res),
                        corte(Resto, Resfinal, Res, []).

corte([Cab|Resto], Resfinal, Resprov, Sublista):-
                        not(signPunt(Cab)),
                        append(Sublista, [Cab], Prov),
                        corte(Resto, Resfinal, Resprov, Prov).
```

```
% Los siguientes números se corresponden con los códigos ASCII de los signos de puntuación

signPunt(32).
signPunt(33).
signPunt(34).
signPunt(38).
signPunt(40).
signPunt(41).
signPunt(44).
signPunt(45).
signPunt(46).
signPunt(47).
signPunt(58).
signPunt(59).
signPunt(63).
signPunt(132).
signPunt(147).
signPunt(148).
signPunt(161).
signPunt(191).

% El procedimiento comprobar_palabras/2  coge cada una de las listas individuales que contienen palabras y las manda al proceso de silabeo, una a una.

comprobar_palabras([],[]).

comprobar_palabras([Z|W],Sobras):-
        silabear(Z,F),
        comprobar_palabras(W,Sobras),
        !.

comprobar_palabras([Z|W],[Z|Sobras]):-
        not(silabear(Z,F)),
        comprobar_palabras(W,Sobras),
        !.

% El proceso de silabeo se inicia con la transformación de lista de código ANSII a letras sueltas para poder silabear, y lo manda ya en forma de letras al proceso auxiliar de silabeo

silabear(X,Z):-
    atom_codes(Y,X),
    atom_chars(Y,W),
    silabear_aux(W,Z).
```

```
% El proceso auxiliar de silabeo comprueba que las combinaciones de las letras dentro
de la palabra siguen la estructura silábica del español que recoge el predicado
posibilidades/4.

silabear_aux([],[]).

silabear_aux([Z|T],[Sílaba|Sílaba_resto]):-
        posibilidades(Z,Silabeo,[Z|T],Resto),
        atom_chars(Sílaba,Silabeo),
        silabear_aux(Resto,Sílaba_resto).

% Vocal: una vocal es vocal si está clasificada  en la base de conocimiento como vocal
fuerte o como vocal débil

vocal(X):-
    vocal_fuerte(X);
    vocal_débil(X).
```

4.- Base de Conocimiento

```
vocal_débil(i).
vocal_débil(u).
vocal_débil(y).
vocal_fuerte(a).
vocal_fuerte(e).
vocal_fuerte(o).
vocal_fuerte(á).
vocal_fuerte(é).
vocal_fuerte(ó).
vocal_fuerte(í).
vocal_fuerte(ú).

cons_f(r).
cons_f(s).
cons_f(l).
cons_f(n).
cons_f(d).
cons_f(z).

consFinalTotal(s).

bilabial(p).
bilabial(b).

dental(t).
```

```
dental(c).


líquida(r).
líquida(l).

% Consontantes emparejadas: aquí aparecen recogidas las combinaciones de letras
que forman parejas.

% b
consPareja(b,[b,Liquida|_]):-
                líquida(Liquida).
% f
consPareja(f,[f,Liquida|_]):-
                líquida(Liquida).

% g
consPareja(g,[g,Liquida|_]):-
                líquida(Liquida).

% c
consPareja(c,[c,Liquida|_]):-
                líquida(Liquida).

% g
consPareja(p,[p,Liquida|_]):-
                líquida(Liquida).

% d
consPareja(d,[d,r|_]).

% t
consPareja(t,[t,r|_]).

% ch
consPareja(c,[c,h|_]).

% ll
consPareja(l,[l,l|_]).

% rr
consPareja(r,[r,r|_]).

% qu
consPareja(q,[q,u|_]).
```

```
% Vocales emparejadas: diptongos. Un diptongo se da cuando dos vocales aparecen
unidas y al menos una de ellas es vocal débil.

diptongo(Vocal,[Vocal,Vocal2|_]):-
                vocal(Vocal),
                vocal_débil(Vocal2).

diptongo(Vocal,[Vocal,Vocal2|_]):-
                vocal_débil(Vocal),
                vocal(Vocal2).

%Consontantes pseudofinales: contextos necesarios para que una consonante sea
pseudofinal

% Letra m
pseudocons_f(m,[m,Bilabial|_]):-
                bilabial(Bilabial).

pseudocons_f(m,[m,n|_]).

% Letra g
pseudocons_f(g,[g,n|_]).

% Letra p
pseudocons_f(p,[p,Dentales|_]):-
                dental(Dentales).

% Letra c
pseudocons_f(c,[c,c|_]).

pseudocons_f(c,[c,t|_]).

% Consonantes
cons(b).
cons(c).
cons(f).
cons(g).
cons(h).
cons(j).
cons(k).
cons(l).
cons(m).
cons(ñ).
cons(p).
cons(r).
cons(t).
cons(v).
```

```
cons(w).
cons(x).
cons(y).

%La Y aparece considerada como vocal y consonante
% La Q sólo está reconocida como parte de una pareja de consonantes, ya que no puede funcionar si no va seguida de U



cons(X):-
      cons_f(X);
      cons_tr(X).
```

5.- Lógica del Español

```
% posibilidades/4: letra que encabeza la sílaba/sílaba/letras que siguen/resto de la palabra (sin la sílaba que acabamos de aislar, pero sí con el contexto).

% PREFIJOS: Combinaciones de letras que forman prefijos y que por lo tanto pueden ir seguidas de cualquier letra.

% sub
posibilidades(s,[s,u,b,s],[s,u,b,s,Cons|W],Resto):-
                                    cons(Cons),
                                    append([s,u,b,s],Resto,[s,u,b,s,Cons|W]),
                                    !.

posibilidades(s,[s,u,b],[s,u,b,Cons|W],Resto):-
                                    cons(Cons),
                                    append([s,u,b],Resto,[s,u,b,Cons|W]),
                                   !.
% ob
posibilidades(o,[o,b,s],[o,b,s,Cons|W],Resto):-
                                    cons(Cons),
                                    append([o,b,s],Resto,[o,b,s,Cons|W]),
                                     !.

posibilidades(o,[o,b],[o,b,Cons|W],Resto):-
                                    cons(Cons),
                                    not(consPareja(b,[b,Cons])),
                                    append([o,b],Resto,[o,b,Cons|W]),
                                     !.

%ab
posibilidades(a,[a,b,s],[a,b,s,Cons|W],Resto):-
```

```
                                    cons(Cons),
                                    append([a,b,s],Resto,[a,b,s,Cons|W]),
                                    !.

posibilidades(a,[a,b],[a,b,Cons|W],Resto):-
                                   cons(Cons),
                                    not(consPareja(b,[b,Cons])),
                                     append([a,b],Resto,[a,b,Cons|W]),
                                    !.

%ex
posibilidades(e,[e,x],[e,x,Cons|W],Resto):-
                                    cons(Cons),
                                    append([e,x],Resto,[e,x|W]),
                                     !.

% sílabas que empiezan por vocal

% a
posibilidades(Vocal,[Vocal],[Vocal],Resto):-
                                    vocal(Vocal),
                                    append([Vocal],Resto,[Vocal]),
                                    !.

% a-la
posibilidades(Vocal,[Vocal],[Vocal,Cons,Vocal1|W],Resto):-
                                    vocal(Vocal),
                                    cons(Cons),
                                    vocal(Vocal1),
                                    append([Vocal],Resto,[Vocal,Cons,Vocal1|W]),
                                    !.

% a-pro-xi-mar-se
posibilidades(Vocal,[Vocal],[Vocal,ConsPareja,Cons_tr,Vocal2|W],Resto):-
          vocal(Vocal),
          consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
          append([Vocal],Resto,[Vocal,ConsPareja,Cons_tr,Vocal2|W]),
          !.

% a-é-re-o
posibilidades(Vocal,[Vocal],[Vocal,Vocal2|W],Resto):-
                                     vocal_fuerte(Vocal),
                                     vocal_fuerte(Vocal2),
                                     append([Vocal],Resto,[Vocal,Vocal2|W]),
                                     !.

% en
```

```
posibilidades(Vocal,[Vocal,Cons_f],[Vocal,Cons_f],Resto):-
                                    vocal(Vocal),
                                    cons_f(Cons_f),
                                    append([Vocal,Cons_f],Resto,[Vocal,Cons_f]),
                                    !.

% is-la
posibilidades(Vocal,[Vocal,Cons_f],[Vocal,Cons_f,ConsBis,VocalBis|W],Resto):-
                                     vocal(Vocal),

(cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsBis|W])),
                                    cons(ConsBis),
                                    vocal(VocalBis),

append([Vocal,Cons_f],Resto,[Vocal,Cons_f,ConsBis,VocalBis|W]),
                                    !.
% antro, an-cla
posibilidades(Vocal,[Vocal,Cons_f],[Vocal,Cons_f,ConsPareja,Cons_tr|W],Resto):-

                vocal(Vocal),
                cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsPareja|_])),
                 consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                 append([Vocal,Cons_f],Resto,[Vocal,Cons_f,ConsPareja,Cons_tr|W]),
                  !.

%ins-tau-rar
posibilidades(Vocal,[Vocal,Cons_f,ConsFinalTotal],[Vocal,Cons_f,ConsFinalTotal,Con
sBis|W],Resto):-
                                   vocal(Vocal),
                                    cons_f(Cons_f),
                                     consFinalTotal(ConsFinalTotal),
                                    cons(ConsBis),

append([Vocal,Cons_f,ConsFinalTotal],Resto,[Vocal,Cons_f,ConsFinalTotal,ConsBis|
W]),
                                    !.

% ay

posibilidades(Vocal,[Vocal,Vocal2],[Vocal,Vocal2],Resto):-
                                    diptongo(Vocal,[Vocal,Vocal2|_]),
                                    append([Vocal,Vocal2],Resto,[Vocal,Vocal2]),
                                    !.

% ai-rar, au-lar, eusebio
posibilidades(Vocal,[Vocal,Vocal2],[Vocal,Vocal2,Cons,Vocal1|W],Resto):-
                                        diptongo(Vocal,[Vocal,Vocal2|_]),
```

```
                                        cons(Cons),
                                        vocal(Vocal1),

append([Vocal,Vocal2],Resto,[Vocal,Vocal2,Cons,Vocal1|W]),
                                        !.

% ai-trar
posibilidades(Vocal,[Vocal,Vocal2],[Vocal,Vocal2,ConsPareja,Cons_tr|W],Resto):-
                                        diptongo(Vocal,[Vocal,Vocal2|_]),
                                        consPareja(ConsPareja,[ConsPareja,Cons_tr]),

append([Vocal,Vocal2],Resto,[Vocal,Vocal2,ConsPareja,Cons_tr|W]),
                                        !.
% a-ma-rí-AIS
posibilidades(Vocal,[Vocal,Vocal2,Cons_f],[Vocal,Vocal2,Cons_f],Resto):-
                                    diptongo(Vocal,[Vocal,Vocal2|_]),
                                    cons_f(Cons_f),

append([Vocal,Vocal2,Cons_f],Resto,[Vocal,Vocal2,Cons_f]),
                                    !.
% AIS-lar
posibilidades(Vocal,[Vocal,Vocal2,Cons_f],[Vocal,Vocal2,Cons_f,Cons|W],Resto):-

                                    diptongo(Vocal,[Vocal,Vocal2|_]),

(cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,Cons|_])),
                                    cons(Cons),

append([Vocal,Vocal2,Cons_f],Resto,[Vocal,Vocal2,Cons_f,Cons|W]),
                                    !.

% ea: hiato aislado.
posibilidades(Vocal,[Vocal],[Vocal,Vocal2],Resto):-
                                    vocal_fuerte(Vocal),
                                    vocal_fuerte(Vocal2),
                                    append([Vocal],Resto,[Vocal,Vocal2]),
                                    !.

% posibilidades para CONS+VOCAL AISLADO
posibilidades(Cons,[Cons,Vocal],[Cons,Vocal],Resto):-
                        cons(Cons),
                        vocal(Vocal),
                        append([Cons,Vocal],Resto,[Cons,Vocal]),
                        !.

% posibilidaddes para CONS+VOCAL EN PALABRA
```

```
posibilidades(Cons,[Cons,Vocal],[Cons,Vocal,Cons1,Vocal1|W],Resto):-
                        cons(Cons),
                        vocal(Vocal),
                        cons(Cons1),
                        vocal(Vocal1),
                        append([Cons,Vocal],Resto,[Cons,Vocal,Cons1,Vocal1|W]),
                        !.

% posibilidaddes para CONS+VOCAL EN PALABRA: HIATO

posibilidades(Cons,[Cons,Vocal],[Cons,Vocal,Vocal2|W],Resto):-
                        cons(Cons),
                        vocal_fuerte(Vocal),
                        vocal_fuerte(Vocal2),
                        append([Cons,Vocal],Resto,[Cons,Vocal,Vocal2|W]),
                        !.

% posibilidades para CONS+VOCAL+VOCAL AISLADO: HIATO
posibilidades(Cons,[Cons,Vocal],[Cons,Vocal,Vocal2],Resto):-
                        cons(Cons),
                        vocal_fuerte(Vocal),
                        vocal_fuerte(Vocal2),
                        append([Cons,Vocal],Resto,[Cons,Vocal,Vocal2]),
                        !.

% CONS+ VOCAL en palabra con consonante pareja: nu-blar  pa-dre
posibilidades(Cons,[Cons,Vocal],[Cons,Vocal,ConsPareja,Cons_tr|W],Resto):-
                        cons(Cons),
                        consPareja(ConsPareja,[ConsPareja,Cons_tr]),
                        vocal(Vocal),
                        append([Cons,Vocal],Resto,[Cons,Vocal,ConsPareja,Cons_tr|W]),
                        !.


% CONS+VOCAL+CONS_FIN EN FINAL DE PALABRA: del-fin
posibilidades(Cons,[Cons,Vocal,Cons_f],[Cons,Vocal,Cons_f],Resto):-
                        cons(Cons),
                        vocal(Vocal),
                        cons_f(Cons_f),
                        append([Cons,Vocal,Cons_f],Resto,[Cons,Vocal,Cons_f]),
                        !.

% CONS+VOCAL+CONS_F: pen-sar
posibilidades(Cons,[Cons,Vocal,Cons_f],[Cons,Vocal,Cons_f,Cons1,Vocal1|W],Resto):
-
                        cons(Cons),
                        vocal(Vocal),
```

```
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,Cons1|_])),
                    cons(Cons1),
                    vocal(Vocal1),

append([Cons,Vocal,Cons_f],Resto,[Cons,Vocal,Cons_f,Cons1,Vocal1|W]),
                    !.

% CONS+VOCAL+CONS_F: pen-trar
posibilidades(Cons,[Cons,Vocal,Cons_f],[Cons,Vocal,Cons_f,ConsPareja,Cons_tr|W],R
esto):-
                    cons(Cons),
                    vocal(Vocal),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsTipo|W])),
                    consPareja(ConsPareja,[ConsPareja,Cons_tr]),

append([Cons,Vocal,Cons_f],Resto,[Cons,Vocal,Cons_f,ConsPareja,Cons_tr|W]),
                   !.

% cons-truir
posibilidades(Cons,[Cons,Vocal,Cons_f,ConsFinalTotal],[Cons,Vocal,Cons_f,ConsFina
lTotal,Cons1|W],Resto):-
                    cons(Cons),
                    vocal(Vocal),
                    cons_f(Cons_f),
                    consFinalTotal(ConsFinalTotal),
                    cons(Cons1),


append([Cons,Vocal,Cons_f,ConsFinalTotal],Resto,[Cons,Vocal,Cons_f,ConsFinalTota
l,Cons1|W]),
                    !.

% posibilidades para CONS+VOCAL+VOCAL AISLADO: diptongo descendente
posibilidades(Cons,[Cons,Vocal,Vocal2],[Cons,Vocal,Vocal2],Resto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    append([Cons,Vocal,Vocal2],Resto,[Cons,Vocal,Vocal2]),
                    !.

% posibilidades para CONS+VOCAL+VOCAL EN PALABRA: diptongo descendente
posibilidades(Cons,[Cons,Vocal,Vocal2],[Cons,Vocal,Vocal2,Cons1,Vocal1|W],Resto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    cons(Cons1),
                    vocal(Vocal1),

append([Cons,Vocal,Vocal2],Resto,[Cons,Vocal,Vocal2,Cons1,Vocal1|W]),
```

```
                    !.
% nai-blar; nai-trar
posibilidades(Cons,[Cons,Vocal,Vocal2],[Cons,Vocal,Vocal2,ConsPareja,Cons_tr|W],R
esto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),

append([Cons,Vocal,Vocal2],Resto,[Cons,Vocal,Vocal2,ConsPareja,Cons_tr|W]),
                    !.

% CONS+VOCAL+VOCAL+CONS_FIN con dipt descendente AISLADO: a-má-bais

posibilidades(Cons,[Cons,Vocal,Vocal2,Cons_f],[Cons,Vocal,Vocal2,Cons_f],Resto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    cons_f(Cons_f),

append([Cons,Vocal,Vocal2,Cons_f],Resto,[Cons,Vocal,Vocal2,Cons_f]),
                    !.

% CONS+VOCAL+VOCAL+CONS_F: DIPT DESCENDENTE mais-car

posibilidades(Cons,[Cons,Vocal,Vocal2,Cons_f],[Cons,Vocal,Vocal2,Cons_f,Cons1,Vo
cal1|W],Resto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,Cons1|_])),
                    cons(Cons1),
                    vocal(Vocal1),
append([Cons,Vocal,Vocal2,Cons_f],Resto,[Cons,Vocal,Vocal2,Cons_f,Cons1,Vocal1|
W]),
                    !.

% CONS+VOCAL+VOCAL+CONS_F: DIPT DESCENDENTE mais-bla
posibilidades(Cons,[Cons,Vocal,Vocal2,Cons_f],[Cons,Vocal,Vocal2,Cons_f,ConsTipo,
Cons_tr|W],Resto):-
                    cons(Cons),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsTipo|_])),
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),

append([Cons,Vocal,Vocal2,Cons_f],Resto,[Cons,Vocal,Vocal2,Cons_f,ConsPareja,Co
ns_tr|W]),
                    !.
% CONS+VOCAL+VOCAL+CONS_F*CONSFINALTOTAL: DIPT DESCENDENTE
mains-bla
```

```
posibilidades(Cons,[Cons,Vocal,Vocal2,Cons_f,ConsFinalTotal],[Cons,Vocal,Vocal2,C
ons_f,ConsFinalTotal,Cons1|W],Resto):-
                        cons(Cons),
                        diptongo(Vocal,[Vocal,Vocal2|_]),
                        cons_f(Cons_f),
                        cons(Cons1),
append([Cons,Vocal,Vocal2,Cons_f,ConsFinalTotal],Resto,[Cons,Vocal,Vocal2,Cons_f
,ConsFinalTotal,Cons1|W]),
                        !.

% bla
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal],[ConsPareja,Cons_tr,Vocal],Res
to):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        vocal(Vocal),

append([ConsPareja,Cons_tr,Vocal],Resto,[ConsPareja,Cons_tr,Vocal]),
                       !.

% gra-pa
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal],[ConsPareja,Cons_tr,Vocal,Con
s,Vocal1|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        vocal(Vocal),
                        vocal(Vocal1),
                        cons(Cons),
append([ConsPareja,Cons_tr,Vocal],Resto,[ConsPareja,Cons_tr,Vocal,Cons,Vocal1|W
]),
                        !.

% pro-e-mio
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal],[ConsPareja,Cons_tr,Vocal,Voc
al2|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        vocal_fuerte(Vocal),
                        vocal_fuerte(Vocal2),
append([ConsPareja,Cons_tr,Vocal],Resto,[ConsPareja,Cons_tr,Vocal,Vocal2|W]),
                        !.

% bla-bla
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal],[ConsPareja,Cons_tr,Vocal,Con
sPareja1,Cons_tr1|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        vocal(Vocal),
                        consPareja(ConsPareja1,[ConsPareja1,Cons_tr1|_]),
append([ConsPareja,Cons_tr,Vocal],Resto,[ConsPareja,Cons_tr,Vocal,ConsPareja1,C
ons_tr1|W]),
```

```
                    !.

% blas
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Cons_f],[ConsPareja,Cons_tr,Vo
cal,Cons_f],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal(Vocal),
                    cons_f(Cons_f),
append([ConsPareja,Cons_tr,Vocal,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,Cons_f]
),
                    !.

% blan-ca
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Cons_f],[ConsPareja,Cons_tr,Vo
cal,Cons_f,Cons1,Vocal1|W],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal(Vocal),
                    vocal(Vocal1),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,Cons1|_])),
                    cons(Cons1),

append([ConsPareja,Cons_tr,Vocal,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,Cons_f,
Cons1,Vocal1|W]),
                    !.

% tran-cla
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Cons_f],[ConsPareja,Cons_tr,Vo
cal,Cons_f,ConsPareja1,Cons_tr1|W],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal(Vocal),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsTipo|_])),
                    consPareja(ConsPareja1,[ConsPareja1,Cons_tr1|_]),
                    cons_tr(Cons_tr2),
append([ConsPareja,Cons_tr,Vocal,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,Cons_f,
ConsPareja1,Cons_tr1|W]),
                    !.
% transs-pa-ren-te

posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Cons_f,ConsFinalTotal],[ConsPa
reja,Cons_tr,Vocal,Cons_f,ConsFinalTotal,Cons1|W],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal(Vocal),
                    cons_f(Cons_f),
                    consFinalTotal(ConsFinalTotal),
                    cons(Cons1),
append([ConsPareja,Cons_tr,Vocal,Cons_f,ConsFinalTotal],Resto,[ConsPareja,Cons_t
r,Vocal,Cons_f,ConsFinalTotal,Cons1|W]),
```

```
                        !.

%
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2],[ConsPareja,Cons_tr,Vo
cal,Vocal2],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        diptongo(Vocal,[Vocal,Vocal2|_]),
append([ConsPareja,Cons_tr,Vocal,Vocal2],Resto,[ConsPareja,Cons_tr,Vocal,Vocal2]
),
                        !.

% brai-le
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2],[ConsPareja,Cons_tr,Vo
cal,Vocal2,Cons,Vocal1|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        diptongo(Vocal,[Vocal,Vocal2|_]),
                        vocal(Vocal1),
                        cons(Cons),
append([ConsPareja,Cons_tr,Vocal,Vocal2],Resto,[ConsPareja,Cons_tr,Vocal,Vocal2,
Cons,Vocal1|W]),
                        !.

% blau-bla
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2],[ConsPareja,Cons_tr,Vo
cal,Vocal2,ConsParejaBis,Cons_trBis|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        diptongo(Vocal,[Vocal,Vocal2|_]),
                        consPareja(ConsParejaBis,[ConsParejaBis,Cons_trBis|_]),
append([ConsPareja,Cons_tr,Vocal,Vocal2],Resto,[ConsPareja,Cons_tr,Vocal,Vocal2,
ConsParejaBis,Cons_trBis|W]),
                        !.

% blaun
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f],[ConsPareja,Con
s_tr,Vocal,Vocal2,Cons_f],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                        diptongo(Vocal,[Vocal,Vocal2|_]),
                        cons_f(Cons_f),

append([ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,
Vocal2,Cons_f]),
                       !.

% brain-le
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f],[ConsPareja,Con
s_tr,Vocal,Vocal2,Cons_f,Cons,Vocal1|W],Resto):-
                        consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
```

```
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    vocal(Vocal1),
                    cons(Cons),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,Cons|_])),

append([ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,
Vocal2,Cons_f,Cons,Vocal1|W]),
                    !.

% blaun-bla
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f],[ConsPareja,Con
s_tr,Vocal,Vocal2,Cons_f,ConsParejaBis,Cons_trBis|W],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    (cons_f(Cons_f);pseudocons_f(Cons_f,[Cons_f,ConsTipoBis|_])),
                    consPareja(ConsParejaBis,[ConsParejaBis,Cons_trBis|_]),

append([ConsPareja,Cons_tr,Vocal,Cons_f],Resto,[ConsPareja,Cons_tr,Vocal,Vocal2,
Cons_f,ConsParejaBis,Cons_trBis|W]),
                    !.
% brains-le
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f,ConsFinalTotal],[
ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f,ConsFinalTotal,Cons|W],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    diptongo(Vocal,[Vocal,Vocal2|_]),
                    cons(Cons),
                    cons_f(Cons_f),
                    consFinalTotal(ConsFinalTotal),

append([ConsPareja,Cons_tr,Vocal,Vocal2,Cons_f,ConsFinalTotal],Resto,[ConsPareja
,Cons_tr,Vocal,Vocal2,Cons_f,ConsFinalTotal,Cons|W]),
                    !.
% Triptongo

posibilidades(Cons,[Cons,Vocal,Vocal2,Vocal3],[Cons,Vocal,Vocal2,Vocal3],Resto):-
                    cons(Cons),
                    vocal_débil(Vocal),
                    vocal_fuerte(Vocal2),
                    vocal_débil(Vocal3),
append([Cons,Vocal,Vocal2,Vocal3],Resto,[Cons,Vocal,Vocal2,Vocal3]),
                    !.

% Triptongo en cons_tr
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Vocal3],[ConsPareja,Con
s_trVocal,Vocal2,Vocal3],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal_débil(Vocal),
```

```
                    vocal_fuerte(Vocal2),
                    vocal_débil(Vocal3),

append([ConsPareja,Cons_tr,Vocal,Vocal2,Vocal3],Resto,[ConsPareja,Cons_tr,Vocal,
Vocal2,Vocal3]),
                    !.

% Triptongo +cons final
posibilidades(Cons,[Cons,Vocal,Vocal2,Vocal3,Cons_f],[Cons,Vocal,Vocal2,Vocal3,Co
ns_f],Resto):-
                    cons(Cons),
                    vocal_débil(Vocal),
                    vocal_fuerte(Vocal2),
                    vocal_débil(Vocal3),
                    cons_f(Cons_f),

append([Cons,Vocal,Vocal2,Vocal3,Cons_f],Resto,[Cons,Vocal,Vocal2,Vocal3,Cons_f]
),
                    !.

% Triptongo en cons_tr + cons final
posibilidades(ConsPareja,[ConsPareja,Cons_tr,Vocal,Vocal2,Vocal3,Cons_f],[ConsPar
eja,Cons_tr,Vocal,Vocal2,Vocal3,Cons_f],Resto):-
                    consPareja(ConsPareja,[ConsPareja,Cons_tr|_]),
                    vocal_débil(Vocal),
                    vocal_fuerte(Vocal2),
                    vocal_débil(Vocal3),
                    cons_f(Cons_f),

append([ConsPareja,Cons_tr,Vocal,Vocal2,Vocal3,Cons_f],Resto,[ConsPareja,Cons_tr
,Vocal,Vocal2,Vocal3,Cons_f]),
                    !.
```

5.-CONCLUSIONES

A la luz de lo expuesto hasta ahora, es fácil ver las aportaciones que supone AZRAEL en el campo de la detección automática del idioma.

1. Aportaciones en el enfoque

AZRAEL es una aproximación nueva al problema de la detección automática del idioma. El modelo teórico en el que está basado el programa es la estructura silábica del español, lo que supone un método distinto al de otros detectores.

Además, aunque la versión de AZRAEL que presentamos está exclusivamente orientada a la detección de textos en español, este modelo de reconocimiento del idioma es aplicable a otras lenguas, lo que ofrece una inmensa cantidad de posibilidades para futuros detectores. De hecho, puesto que la estructura silábica permite distinguir unas lenguas de otras, se abre un prometedor campo de investigación en Tipología Lingüística, ya que sería posible llegar, mediante comparación de lenguas, a una clasificación tipológica de las lenguas en base a su estructura silábica. En este sentido, AZRAEL aúna dos ramas de la Lingüística: por un lado, como herramienta para el tratamiento automático del texto, la Lingüística Computacional y el Procesamiento de Lenguaje Natural; por otro lado, como estudio que busca elementos que caractericen a una lengua, la Tipología Lingüística. Ambas disciplinas tienen mucho que aportarse y esperamos que este trabajo impulse líneas de investigación comunes.

2. Estudio teórico de la sílaba en español.

2.1 Estructura silábica en español

Como no ha sido posible encontrar tratados exhaustivos de la sílaba es español, hemos realizado nuestro propio estudio del tema, cuyas conclusiones aquí resumimos a partir de lo expuesto en el apartado Datos. Recordamos que este estudio silábico está diseñado en base

al texto escrito, y que por lo tanto se consideran letras distintas aquellas que tengan caracteres distintos. Por ello es que la CH la consideramos una combinación consonántica (al igual que BR-) y no una letra. De igual manera, como no se ha realizado el estudio en base a aspectos fonéticos sino exclusivamente gráficos, la letra H recibe el mismo tratamiento que cualquier otra consonante.

La combinación que todas las consonantes admiten es la estructura CONSONANTE+VOCAL, salvo la letra Q, que como veremos más adelante, es especial.

Las consonantes intercaladas la R y la L, y son aquellas que se sitúan entre la consonante inicial de la sílaba y la vocal siguiente.

No todas las consonantes admiten ir acompañadas de consonantes intercaladas. Aquellas consonantes que admiten que las acompañe indistintamente la R o la L, son las *consonantes tipo 1*, y son la B, la C, la F, la G y la P. Las *consonantes tipo 2* son aquellas que sólo admiten como compañera a la R, y son la T y la D.

No es casual que sean justo estas dos letras las que pueden intercalarse: los sonidos a los que ambas letras remiten son sonidos líquidos, que están a medio camino entre los sonidos consonánticos más puros (oclusivos y fricativos) y los vocálicos. Son sonidos que presentan características espectrográficas semejantes a las de una vocal, y que pueden constituir por sí solas núcleo silábico en otros idiomas. De hecho, en gramáticas de algunas lenguas como el sánscrito la L y la R son consideradas vocales. Por lo tanto, existe una explicación fonética para que sean estas dos letras y no otras las que pueden formar combinaciones con otras consonantes.

Las vocales pueden encabezar solas la sílaba, pero si van precedidas de una consonante o grupo consonántico se unen a ellas, como en *to-ro* y *clo-ro*. Si una *vocal fuerte* (A, E, O, Í, Ú) va seguida de una *vocal débil* (I,U) ambas tienden a unirse, formando parte de la misma sílaba (diptongo descendente). En el caso inverso (vocal débil seguida de fuerte, diptongo ascendente) o si ambas vocales son débiles también permanecen unidas. Sin embargo, dos vocales fuertes juntas (hiato) forman parte de sílabas distintas.

Si bien todas las consonantes pueden ser principio de sílaba, no todas pueden ser final de sílaba. Sólo las siguientes consonantes pueden desempeñar esta función sin restricciones: R, S, L, N, D, Z. Son las *consonantes finales*, que pueden no sólo ser final de sílaba sino también final de palabra. Las consonantes finales sólo son final de sílaba si no van seguidas de vocal. En este caso, la consonante final abandonaría la sílaba precedente para unirse a la vocal de la sílaba siguiente y convertirse en la consonante que encabeza la sílaba. Las vocales no tienen restricciones para ser final de sílaba.

Las consonantes pseudofinales que pueden comportarse como consonantes finales, pero sólo en determinados contextos, es decir, si van seguidas de una consonante en concreto. Presentamos a continuación una tabla que recoge las consonantes que pueden ser ocasionalmente final de sílaba.

CONSONANTE PSEUDOFINAL	CONTEXTO CONSONÁNTICO	EJEMPLOS
C	t	ac-tuar
	c	ac-ce-der
G	n	ig-no-rar
M	p	pom-pa
	b	bom-ba
	n	a-lum-no
P	t	a-cep-tar
	c	op-ción

Adicionalmente, nos encontramos con algunas letras que tienen un patrón de comportamiento particular:

- La S es consonante final total ya que puede ir cerrando una sílaba detrás de una consonante final o a un prefijo, ya que es la única consonante que tiene la capacidad de unirse a una consonante final para acabar la sílaba. La mayoría son términos heredados del latín (aunque no todos, como en *sáns-cri-to*), y de hecho muchos tienden a simplificar el grupo consonántico perdiendo la consonante final y conservando sólo la S. Así, por ejemplo, la forma *obs-cu-ro* ha

ido perdiéndose a favor de *os-cu-ro*. Esto nos demuestra que los hablantes perciben como anómalas estas combinaciones consonánticas y tienden a simplificarlas para adaptarlas a la estructura silábica habitual del español. Como en el caso de los prefijos, las palabras que nos llegan con esta estructura son fósiles que nos aportan información sobre su origen y nos revelan cómo era la estructura silábica en otras lenguas.

- La Y la hemos considerado consonante y vocal, porque puede desempeñar un papel u otro según el contexto silábico en el que se encuentre.

- La Q sólo puede funcionar si va seguida de una U, lo que quiere decir que por sí sola no es una letra autónoma e independiente.

- La LL, la RR y la CH son combinaciones de dos letras que representan un único sonido, pero que a diferencia de la Q, pueden funcionar también solas. Son las *consonantes emparejadas*.

Estas son las reglas básicas de la estructura silábica del español. Sin embargo, existen dos casos en los que estas reglas no son aplicables: los prefijos y los cultismos.

2.2 El funcionamiento de los prefijos

Gracias al estudio silábico del español hemos descubierto que en el caso de los prefijos, las reglas de la derivación morfológica se superponen a las reglas de la estructuración silábica. Como los prefijos son partículas y aportan significado por sí mismos, los hablantes admiten que puedan ir seguidos por cualquier consonante. Es decir, aunque los prefijos puedan ser sílabas, las reglas silábicas no se aplican en este caso, ya que los prefijos sin ser independientes, tienen significado autónomo. Este fenómeno se da con los prefijos *sub-* y *ex-*, pero también se aplican a las palabras con las partículas *ab-* y *ob-* al comienzo, ya que ambas eras prefijos en latín, y por lo tanto, en el momento de su creación estabas regidas por las mismas reglas de prefijación que actualmente rigen *ex-* y *sub-,* ya que de igual manera que los hispanohablantes no tienen problema en admitir cualquier consonante detrás de los prefijos *ex-* y *sub-*, los latinos podían construir palabras en las que *ab-* y *ob-* fueran seguidos de cualquier combinación de letras. Aunque las palabras con *ab-* y *ob-*

como prefijos son abundantes, todas provienen del latín y no existen palabras de este tipo que sean de creación posterior. Esto quiere decir que en el momento en que los hablantes perdieron la percepción de *ab-* y *ob-* como prefijos, dejaron de crearse palabras nuevas con esta forma, lo que supone que la posición de B como final de sílaba era percibida como extraña. Las palabras que ya existían se conservaron y han llegado hasta nosotros como fósiles que muestran lo que en otro tiempo sí había sido percibido como admisible, pero esa estructura, una vez que dejó de hablarse el latín, no caló en el patrón silábico del español. Este hecho aporta una gran cantidad de información etimológica, ya que podemos afirmar que todas las palabras que tengan el prefijo *ab-* y *ob-* vendrán del latín. Esto quiere decir que sólo estudiando la estructura de la sílaba podemos obtener una gran cantidad de información etimológica, ya que todas las palabras que empiecen por *ab-* y *ob-* derivarán del latín. De hecho, si mantienen el prefijo intacto quiere decir que mantienen una estructura muy semejante a la que tenían en latín y que la forma apenas se ha visto modificada. Esto nos indica que estas palabras han llegado al castellano por la vía culta, ya que los procesos de derivación y adecuación al español que han sufrido han sido mínimos.

2.3 Cultismos y extranjerismos. Información etimológica.

Hasta aquí hemos descrito la estructura silábica del español. Sin embargo, es posible encontrar palabras españolas que se salgan de este patrón. Las combinaciones consonánticas más frecuentes que son ajenas a la estructura silábica del español son TL, TM, TN, PN, FT, GM, GD, CN y IT. Si analizamos de dónde provienen las palabras que contienen estas combinaciones, nos daremos cuenta de que todas son cultismos o extranjerismos. Como se ve en la siguiente tabla, en la mayoría de los casos son palabras del griego, aunque también hay términos heredados del náhuatl y del árabe.

ORIGEN ETIMOLÓGICO DE COMBINACIONES DE LETRAS		
COMBINACIONES	ORIGEN	EJEMPLOS
TL	GRIEGO O NÁHUATL	atleta, tlacoyo
TM	GRIEGO	Atmósfera
TN	GRIEGO	Etnia
PN	GRIEGO	Hipnosis
FT	GRIEGO O ÁRABE	difteria, muftí
GM	GRIEGO	Diafragma
GD	GRIEGO	Amígdalas
CN	GRIEGO	Arácnido
IT	LATINISMO	Déficit

En ningún caso es un inconveniente el hecho de que estas palabras se salgan de la estructura silábica del español, ya que este hecho nos proporciona valiosa información sobre el origen de estas palabras. De hecho, este fenómeno demuestra la solidez de las reglas de la sílaba: si una palabra no tiene la estructura silábica esperada es porque necesariamente viene de otra lengua.

No es solamente el idioma de procedencia el hecho que determina el aspecto de una palabra importada. Cuanto más reciente haya sido la incorporación, más se alejará de la estructura silábica del español; según vaya transcurriendo el tiempo, la forma de la palabra irá perdiendo las combinaciones anómalas y se irá adecuando a la estructura del español, de manera semejante al proceso en que las piedras de un río se van erosionando por el paso del tiempo y por efecto de la corriente hasta llegar a ser cantos rodados. De este modo, las palabras de origen griego que entraron al español a través del latín están ya adaptadas a la estructura silábica del español, en primer lugar porque el latín ya realizó una primera adaptación de la forma griega a la estructura silábica del latín, y en segundo lugar, porque estas palabras llevan siglos incorporadas al español. Es decir, son los cantos rodados que ya han sufrido el proceso de erosión. En contraposición, las palabras del griego que entraron como cultismos científicos a partir del siglo XVIII tienen una estructura anómala, porque han pasado directamente al español sin el latín como intermediario, y porque su entrada es reciente: son las piedras que acaban de ser incorporadas al cauce del río y aún conservan los picos e irregularidades que irá borrando el paso del tiempo.

Otro factor que influye en la adaptación de una palabra extranjera es la vía por la que llegue al español. Si la palabra está restringida al ámbito culto y académico es más probable que conserve las estructuras anómalas que si la palabra pasa al habla popular. Un ejemplo de este fenómeno son los dobletes castellanos (una palabra culta y otra popular) que derivan de un mismo término latino: por ejemplo, la palabra latina *episcopatus* nos ha dado *obispado* por la vía patrimonial y *episcopado* (mucho más parecida a la forma origina) por la vía culta.

A partir de lo expuesto, con un estudio más detallado sobre el tema (pero que se sale de los objetivos de esta investigación), podríamos llegar a determinar a partir de la forma la época aproximada de incorporación de una palabra al español.

3. Creación de AZRAEL

Basándonos en el estudio teórico que acabamos de exponer, hemos creado AZRAEL, un reconocedor automático de textos en español, y los dos programas auxiliares de los que se sirve: el Silabeador y ESDRA (Etimólogo Selectivo Del Reconocedor Automático). El Silabeador es un programa completo y eficaz de silabeo de palabras, construido sobre el estudio teórico de la estructura silábica del español y que puede usarse de forma independiente a AZRAEL.

AZRAEL, a su vez, es un detector tanto de textos como de palabras sueltas que decide si algo es español o no a partir de la información que le suministra el silabeador, es decir, es un reconocedor morfológico. Una de sus ventajas es que no decide de forma global si un texto es entero español o no, sino que analiza palabra a palabra, y devuelve en una lista aparte lo que no tiene estructura de español. AZRAEL en compañía de ESDRA va un paso más allá, y, además de devolver las palabras que no siguen la estructura silábica del español, asigna posibles orígenes etimológicos a aquellas palabras que sean extranjerismos o cultismos. Los tres programas han sido probados con diversos textos y son eficaces

4. Valoración final

AZRAEL supone un avance tanto desde el punto de vista práctico como en el teórico. En el práctico, hemos logrado crear un programa de detección automática eficaz y completo que además se basa en el mismo mecanismo que seguimos los hablantes para reconocer un idioma: la forma. Detecta, además, tanto textos como palabras, y admite construcciones poco convencionales (como palabras inventadas o jerga callejera) que otros detectores no reconocerían, aun cuando son producciones de los hablantes.

En el aspecto teórico, el estudio previo de la estructura silábica en español en el que está basado el programa nos aporta información nueva sobre la forma de las palabras en español y nos ayuda a comprender mejor porque admitimos determinadas construcciones y rechazamos otras.

De cara al futuro, el modelo en el que está basado AZRAEL permite seguir el mismo método para construir detectores que estén basados en la estructura silábica de otros idiomas. El conocimiento que supondría buscar los patrones silábicos en torno a los cuales se organizan las lenguas sería un filón para los estudios comparativos.

Por otro lado, en esta versión que presentamos, ESDRA es un programa anexo a AZRAEL, pero las posibilidades que ofrece podrían explotarse para hacer un etimólogo autónomo que buscase posible orígenes de las palabras en base a la forma que tengan.

El estudio de la sílaba en español es un aspecto del idioma en el que no se ha investigado lo suficiente: normalmente se salta del estudio de los sonidos de una lengua al estudio de los procedimientos morfológicos de las palabras, pasando por alto el nivel de la sílaba que, como hemos comprobado, es fundamental. Este campo puede enfocarse tanto desde la rama más teórica (estudios diacrónicos, análisis morfológico y etimológico, etc) como desde la aplicada (Procesamiento de Lenguaje Natural y tratamiento automático de texto) y sin duda abrirán las puertas a prometedoras líneas de investigación.

6.-BIBLIOGRAFÍA

6.1-Documentos impresos

ALCARAZ VARÓ, ENRIQUE; MARTÍNEZ LINARES, ANTONIA (2004): *Diccionario de Lingüística moderna*, Barcelona, Ariel.

ÁLVAREZ, ALFREDO I. (2005), *Hablar en español: la cortesia verbal. La pronunciacion del español estándar. Las formas de expresion oral,* Oviedo, Ediciones Nobel.

BENIERS, ELISABETH (ed.) (2000): *Lecturas de morfología.* México. Instituto de Investigaciones Filológicas.

CANO, RAFAEL (coord.) (2004), *Historia de la lengua española*, Barcelona, Ariel.

CHOMSKY, NOAM, MILLER, GEORGE A. (1976): *El análisis formal de los lenguajes naturales*, Madrid, Alberto Corazón, D.L.

HÁLA, BOHUSLAV (1973): *La sílaba. Su naturaleza, su origen y sus transformaciones*, Madrid, Consejo Superior de Investigaciones Científicas.

HARRIS, JAMES W. (1983): *Syllable Structure and Stress in Spanish: A Nonlinear Analysis*, Boston, The Massachusetts Institute of Technology.

HUTCHINS, JOHN W. (2000), *Early years in machine translation. Memoirs and biographies of pioneers*, Philadelphia, John Benjamins Publishing Company.

LÓPEZ GARCÍA, ÁNGEL (2000), *Cómo surgió el español*, Madrid, Gredos.

MENÉNDEZ PIDAL, RAMÓN (2005), Historia de la Lengua española, Madrid, Fundación Menéndez Pidal.

REAL ACADEMIA ESPAÑOLA (1973), Esbozo de una nueva gramática de la lengua española, Madrid, Espasa.

VÄÄNÄNEN, VEIKKO (1988), *Introducción al latín vulgar*, Madrid, Gredos.

VARELA ORTEGA, SOLEDAD (2005), *Morfología léxica: la formación de palabras*, Madrid, Gredos.

6.1-Textos electrónicos

Gerdemann, Dale, "Evaluation of Language Identification Methods", disponible en: http://www.sfs.nphil.uni-tuebingen.de/iscl/Theses/kranig.pdf

GREFENSTETTE, GREGORY (1995), "Comparing two languages identification schemes", en *JADT 3rd Interantional conference on Statistical Analysis of Textual Data*, Roma. Disponible en: http://www.xrce.xerox.com/Publications/Attachments/1995-012/Gref---Comparing-two-language-identification-schemes.pdf

PENA, JESÚS (2008) "El cambio morfológico en el interior de las series de derivación" en *Revista de Investigación Lingüística*, Vol 11, No 1, Universidad de Murcia. Disponible en: http://revistas.um.es/ril/article/view/53771/51791