UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

SISTEMA DE EXTRACCIÓN DE INFORMACIÓN DE CONSULTAS EN LENGUAJE NATURAL

PROYECTO FIN DE CARRERA

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN: ESPECIALIDAD TELEMÁTICA

Autor: Sergio Acebrón García

Tutor: Julio Villena Román

Resumen

Este proyecto ha sido desarrollado con el fin de mejorar la comunicación entre humanos y máquinas. Se trata de un proyecto enmarcado dentro del campo de la tecnología lingüística y más concretamente dentro de las tecnologías de procesamiento de lenguaje natural.

El objetivo que se persigue es que el sistema transforme una oración en lenguaje natural a una forma más fácilmente entendible para el ordenador. Para ello se ha diseñado un algoritmo versátil y eficiente que permite traducir una consulta en texto libre a una salida más estructurada en forma de tuplas objeto-atributo-valor con la información contenida en la consulta expresada por el usuario.

Para la realización de este proyecto han sido muy importantes los conceptos teóricos. Por ello a lo largo de la memoria se irán exponiendo dichos conceptos teóricos. Como complemento a la teoría se ejemplificarán algunas partes para favorecer la comprensión.

Hay un apartado exclusivamente dedicado a cómo funciona esta aplicación, en el cual se analizan los resultados de los ejemplos.

Agradecimientos

Al fin he llegado a este último paso para acabar la carrera. Digamos que el trayecto no ha sido fácil ni rápido y es por todo ello por lo que le quiero agradecer a tantas personas el haber estado ahí durante todo este tiempo.

En primer lugar quiero dar las gracias al que ha estado con este proyecto conmigo, mi querido tutor. Él sabe muy bien que esto no podía haber sido realizado sin él. Por todo esto y por lo que me has enseñado, muchas gracias.

La siguiente persona es Sonia, sin ella las cosas no hubiesen podido hacerse, ya que es la que más fuerza me daba en los momentos de flaqueza. Gracias porque siempre has estado ahí cuando te necesitaba y cuando no también.

Por supuesto mis padres, hermana, abuelos, y resto de familia por el continuo interés y apoyo que han mostrado a lo largo de estos años. Muchas gracias.

Y ahora empecemos con la parte de los amigos, no me imagino como hubiese sido la carrera sin todas las personas que han estado compartiendo de primera mano lo que es dar una clase en la universidad, simplemente se que con vosotros todo es posible. Por eso Son, San, Jan, Juanal, Dave, Alv, Guille, Pepelu, Jorge, Ana, Rachel muchas gracias por todo.

Y por último y no menos importante los amigos de toda la vida, los únicos que te evaden de cualquier situación difícil. Haciendo la vida más divertida. Gracias a Rodri, Marta, Antonio, Tuko, Santy, Rober, Morato, Laura y Patri.

Creo que deben un reconocimiento todos los profesores que me han impartido enseñanza durante toda mi formación, ya que en cierta medida son ellos los partícipes de que yo haya llegado hasta aquí.

Índice del documento

Índice de imágenes

Índice de tablas

1 Introducción

1.1 Marco del proyecto

Este proyecto está dentro del campo de la Inteligencia Artificial. La Inteligencia Artificial (IA) es una descendencia de la informática que está presente en nuestros días, se muestra en muchos de los productos de uso diario, eso sería en cuanto a una visión particular. Sin embargo, en el ámbito de la industria, la IA tiene una labor cada vez más importante, sobre todo en la monitorización de los sistemas, el ahorro energético y la seguridad entre otros. En el ámbito doméstico la domótica, es la parte de la IA que intenta mejorar el confort de las personas.

La sociedad actual, desde hace unos años, tiene una tendencia muy clara a la informatización. Esto está provocando que aumente también el uso de IA dentro de la informática, puesto que consigue automatizar ciertos trabajos gracias a lo cual se facilitan las tareas tanto cotidianas como informáticas. Este proyecto se va a basar en una de las áreas de la IA, la denominada Ingeniería Lingüística, o tecnología del lenguaje, basada en aplicar el conocimiento sobre lenguaje natural, para intentar generar lenguaje humano en todas sus formas. La ingeniería lingüística aprovecha el conocimiento informático del procesamiento del lenguaje natural y del marco lingüístico (traducción, terminología, lingüística computacional…). Dentro de esta disciplina de estudio, existe un amplio número de utilidades como pueden ser los traductores automáticos, gestores de terminología, lematizadores,…

Este campo, en un nivel inferior, también abarca la Recuperación de Información (*information retrieval*), dada la necesidad de información del usuario, se pretende analizar los recursos de información disponibles localizando e interpretando los datos para la resolución del problema que plantea. Es decir, recuperar la información textual que satisfaga la necesidad de información del usuario. En la recuperación de Información Textual, las técnicas de procesado del lenguaje natural son muy utilizadas tanto para facilitar la descripción del contenido de los documentos, como para representar la consulta formulada por el usuario.

1.2 Objetivos

El principal objetivo del presente proyecto consiste en diseñar y desarrollar un sistema que pueda devolver una respuesta estructurada o etiquetada de fácil compresión para el ordenador a partir de una oración en lenguaje natural, en el caso de este proyecto se tratará el idioma español.

Para llevar a cabo esta tarea se realizará un estudio y análisis de las técnicas actuales de los sistemas de procesamiento de lenguaje natural.

Por otra parte, y referente a la funcionalidad e implementación del sistema se cumplirán los siguientes requisitos:

- El sistema estará centrado en el español, aunque en la medida de lo posible será multilingüe.
- La interfaz de usuario debe de ser lo más intuitiva posible, para así facilitarle su uso al usuario y debe de estar bien organizada para poder visualizar de manera más rápida el resultado general, siendo así mucho más llamativa y atractiva de utilizar.
- El sistema será versátil, modular y portable, para posibles incorporaciones futuras en proyectos más grandes, o incluso en una interfaz Web, donde tendría un entorno más "amigable".
- El tiempo que el sistema tarde en buscar los datos y devolver una respuesta correcta será válido para una aplicación en tiempo real.

1.3 Contenido de la memoria

La estructura de este documento se divide en los siguientes apartados:

- **Capítulo 1:** El capítulo actual, se muestra una visión global sobre el proyecto en sí, así como una visión desglosada sobre los objetivos que se pretenden lograr con este sistema.
- **Capítulo 2:** En este apartado, se expondrá la situación actual de los principales conceptos relacionados con la funcionalidad de la aplicación. Se comenzará explicando

qué es la ingeniería lingüística, y las premisas de las que parte, acercando al lector términos que tendrán que ver a la hora de resolver los problemas relacionados con el lenguaje. Se continuará presentando qué es XML, sus características, y un ejemplo de su estructura.

- **Capítulo 3:** Este capítulo se corresponde con el diseño, por lo que se comentará cómo se ha creado la aplicación, los pasos que sigue hasta que se genera una respuesta, así como las llamadas entre los elementos del sistema.
- **Capítulo 4:** Este capítulo explicará los diferentes algoritmos que se han implementado para llevar a cabo el desarrollo de la aplicación.
- **Capítulo 5:** Este capítulo muestra ejemplos sobre cómo funciona la aplicación, seleccionando distintos parámetros para que se observen las diferencias.
- **Capítulo 6:** En este capítulo se mostrará el presupuesto teórico de este proyecto, desglosado en cada uno de sus apartados.
- **Capítulo 7:** Tras el trabajo realizado, se ha llegado a unas conclusiones que son las aquí expuestas, además de posibles formas de mejorar el producto y nuevas líneas de investigación para ampliar su funcionalidad.
- **Capítulo 8:** Este capítulo es un anexo con todas las listas de palabras usadas para el desarrollo del proyecto.
- **Capítulo 9:** Se mostrarán cada una de las referencias que se han utilizado y sobre las que se apoyan las bases de este documento.

2 Estado del arte

Este capítulo tiene como finalidad enumerar y explicar las bases técnicas que han sido necesarias para el desarrollo del proyecto.

En el primer subcapítulo se va a tratar la Ingeniería Lingüística, su definición, las necesidades de esta ingeniería y el lenguaje y los diversos problemas de la lingüística en el procesamiento del lenguaje natural.

El procesamiento del lenguaje natural es una de las partes fundamentales de este proyecto, por lo que se explicará su arquitectura típica, la segmentación, detección de entidades y etiquetados y análisis sintáctico y morfológico.

Por último, puesto que el desarrollo del sistema se ha realizado en el lenguaje estructurado XML, también se dedicará un subcapítulo de este estado del arte a explicar las bases de este lenguaje.

En el desarrollo de estos subcapítulos se empleará el uso de diversos ejemplos para facilitar la comprensión de los temas descritos.

2.1 Ingeniería lingüística

El término de Ingeniería lingüística abarca un amplio espectro de actividades que suelen englobarse dentro de lo que se ha denominado “las industrias de la lengua”. Comprende una serie de técnicas relacionadas con el tratamiento informático del lenguaje, en este caso del lenguaje natural. El lenguaje natural es el lenguaje que usamos los humanos para comunicarnos y expresarnos.

Este tratamiento informático del lenguaje solía dividirse entre las técnicas que se aplicaban al lenguaje hablado y las propias del procesamiento del texto escrito, pero existe cada vez una mayor convergencia entre ambos métodos. Lo que se persigue es mejorar la interoperabilidad de los sistemas informáticos. Se consigue perfeccionar la interacción de los usuarios con los sistemas informáticos, iteración hombre-máquina más adecuado; y además, que el sistema sea capaz de asimilar, seleccionar, utilizar y presentar la información en función de las necesidades de la aplicación que se desee darle.

2.1.1 Definición

La Ingeniería Lingüística según la Wikipedia [1] “es una disciplina también denominada Informática aplicada a la Lingüística e incluso Tecnología del Lenguaje, que, a su vez, tiene carácter multidisciplinar. La ingeniería lingüística se aprovecha del conocimiento desarrollado en el marco informático del procesamiento del lenguaje natural y del marco lingüístico nutrido por las disciplinas de la traducción, de la terminología y de la lingüística computacional, tanto en sus vertientes teóricas como aplicadas. ”

En definitiva, se podría definir la Ingeniería Lingüística como la tecnología encargada del lenguaje natural, en cuanto a procesamiento y lo que ello conlleva, así como sus posibles aplicaciones.

2.1.2 El lenguaje y las necesidades de la ingeniería lingüística

El lenguaje puede ser entendido como un recurso que hace posible la comunicación. En el caso de los seres humanos, es una herramienta que se encuentra extremadamente desarrollada, que brinda la posibilidad al hombre de selección, citar, coordinar y combinar conceptos de diversa complejidad.

Existen numerosas maneras del lenguaje. El concepto de lengua natural o lenguaje natural, describe a una modalidad lingüística o tipo de lenguaje que el hombre desarrolla con el propósito de comunicarse con su entorno. Esta herramienta, según se advierte al analizar sus particularidades, posee sintaxis y tiene su base en los preceptos de optimización y economía. Además el leguaje ayuda a crear una cultura y se transmite de generación en generación, es decir, está ligado al concepto de sociedad.

Se vive en un mundo en el que el acceso a la información es cada vez más una necesidad, esta necesidad se ha visto impulsada por el desarrollo de las nuevas tecnologías. Éstas están al alcance de cualquier usuario que esté dispuesto a usarlas y ofrecen una gran cantidad de datos y de modos de interacción, gracias a Internet podemos hablar con cualquier persona sin importar la distancia, comprar, tener información de todo tipo. Todo esto lo recoge el concepto de sociedad de la información, aquella en la cual las tecnologías facilitan la creación, distribución y manipulación de la información juegan un papel importante en las actividades sociales, culturales y económicas.

Esta sociedad de la información tiene sus inconvenientes, entre otros el exceso de información que existe actualmente en la red, en estos tiempos acceder a todo tipo de información es relativamente fácil, pero se ha de disponer de los medios para obtener exclusivamente la información de lo que se desea saber de esa información.

Debido a este nuevo entorno que se ha creado, las nuevas tecnologías deben dar un acceso a la información que sea rápido, sencillo y eficaz.

Así el desarrollo de aplicaciones se tendrá que enfocar en crear programas que procesen el lenguaje, para ello se tendrán que generar algoritmos y funcionalidades. Todo ello teniendo en cuenta el gran volumen de información que se va a tratar y contemplar los posibles errores que puedan introducir los usuarios.

A consecuencia de todo lo expuesto surge la problemática de la Recuperación de Datos, implica que esta ingeniería desarrolle aplicaciones capaces de analizar los recursos de información disponibles localizando e interpretando los datos para la resolución del problema que plantea. Es decir, recuperar la información textual que satisfaga la necesidad de información del usuario. Facilitando la descripción del contenido de documentos así como para representar las consultas formuladas por el usuario.

Procesamiento del lenguaje natural. Niveles del lenguaje

El lenguaje natural, como se ha descrito anteriormente, es la modalidad lingüística que el hombre ha desarrollado para comunicarse con su entorno; por tanto, dentro de la ingeniería lingüística existe una especialidad que se encarga del estudio de este lenguaje. Las labores de esta especialidad son la de construir sistemas y mecanismos que permitan o faciliten la comunicación entre personas y máquinas para así facilitar la búsqueda de información. Uno de los grandes retos de la informática es el desarrollo de ordenadores que sean capaces de entender el lenguaje natural.

El lenguaje natural, o mejor dicho, la sintaxis del lenguaje natural se corresponde a un lenguaje formal, similar a los lenguajes lógicos y matemáticos, con lo que se permite que pueda ser modelado.

Para estudiar la forma en la que se estructura el lenguaje natural se utiliza lo que se denominan, tradicionalmente, niveles del lenguaje. En total existen cinco niveles del lenguaje que se detallan a continuación:

- **Fonología**
- **Morfología**
- **Sintaxis**
- **Semántica**
- **Pragmática**

Fonología

La fonología [2] es una disciplina basada en el estudio de los fonemas o segmentos mínimos de la corriente fónica que tienen entidad como elementos del sistema lingüístico, teniendo en cuenta su valor distintivo y funcional. El número de fonemas de una lengua oscila entre veinte y cuarenta.

No hay que confundir la fonología con la fonética, ya que esta última se encarga de estudiar la naturaleza acústica de los sonidos, es decir, la articulación de los sonidos. Mientras que la fonología describe el modo en que los sonidos funcionan, en una lengua en particular o en las lenguas en general, en un nivel abstracto o mental.

Dentro de este campo hay que definir lo que son los pares mínimos, se trata de palabras distintas con significado diferente y que sólo varían en un sonido. Un par de ejemplos para entender los pares mínimos pueden ser: *casa* y *masa*, o *boca* y *roca*.

Como hemos visto los sonidos que componen una palabra son las unidades mínimas que la hacen diferente de otra, es decir, que hay unidades mínimas que diferencian los significados, los fonemas [3].

Los fonemas se definen siguiendo unas normas físicas y articulatorias, en función de su carácter sonoro o sordo. Como ejemplos de tipos de sonidos que se pueden encontrar en una lengua están: bilabial, nasal, consonántico, etc.

Entre los principales rasgos fonéticos que se tienen en cuenta para distinguir fonemas aparecen su **consonanticidad**, su **silabicidad**, su **sonoranticidad**, su **sonoridad** y **aspiración**, su **modo de articulación** y su **punto** o **lugar de articulación**.

Para las aplicaciones de audio o vídeo, como son: los programas de escritura predictiva, programas que generan audio a partir de texto, etc. la fonología es una parte del lenguaje que hay que tener en cuenta. Sobre todo, teniendo en cuenta la cantidad de ambigüedades que pueden aparecer debido al tipo de entonación, letras mudas, fonemas parecidos entre palabras, etc.

Los problemas en fonética computacional están conectados al desarrollo de sistemas de análisis y síntesis del habla. Aun cuando hay sistemas de reconocimiento de voz, el porcentaje de palabras identificadas correctamente es todavía bastante bajo. Entre sistemas de generación de voz hay mucho más progreso, basados en síntesis compilativas, aunque su área de aplicación es bastante restringida.

Morfología

La morfología [4] es una rama de la lingüística que se encarga de estudiar la estructura interna de las palabras.

Trata de delimitar, definir y clasificar las clases de palabras a las que da lugar (morfología flexiva) así como la formación de nuevas palabras (morfología léxica).

La historia de la morfología data del siglo XIX y únicamente en sus inicios trataba la forma de las palabras, actualmente su acepción es estudiar fenómenos más complejos que simplemente la forma.

Un ejemplo claro de morfemas es el siguiente: pato, la palabra se forma con dos monemas.

El lexema (raíz de la palabra), es *pat* y los morfemas (prefijos, sufijos o desinencias) son {*-o -a -os -as*} dando lugar a las palabras: *pato, pata, patos, patas*.

Por tanto, del anterior ejemplo se puede sacar que dado un conjunto de palabras que comparten un mismo lexema, éste se puede denominar raíz.

Dentro del campo de las aplicaciones informáticas la morfología se usa para descomponer y etiquetar las palabras para así hacer el análisis de una palabra. Con esto se consiguen programas tales como correctores ortográficos o gramaticales. Los problemas de la morfología computacional están relacionados con el desarrollo de los sistemas de análisis y síntesis automático morfológico. El desarrollo de tales módulos es bastante difícil porque hay que hacer grandes diccionarios de raíces, en general existe la metodología para tal desarrollo y existen sistemas funcionando para muchos idiomas.

Sintaxis

La sintaxis [5] es la parte de la gramática que estudia la forma en que se combinan y relacionan las palabras para formar secuencias mayores, cláusulas y oraciones.

Hay que destacar que el análisis sintáctico de una oración supone la búsqueda del verbo conjugado y así llegar a distinguir entre el sintagma sujeto y el sintagma predicado. Una forma sencilla de identificar estos sintagmas es una vez ubicado el verbo, preguntar quién o qué realiza la acción. La respuesta a esa pregunta, es el sujeto mientras que el resto es el predicado.

La sintaxis es la parte de la gramática que estudia las reglas que gobiernan o rigen la combinatoria de constituyentes sintácticos y la formación de unidades superiores a estos, como los sintagmas y oraciones gramaticales. La sintaxis, por tanto, estudia las formas en que se combinan las palabras, así como las relaciones sintagmáticas y paradigmáticas existentes entre ellas.

La sintaxis computacional debe tener métodos automáticos para análisis y síntesis, es decir, para construir la estructura de la frase por la frase, o generar la frase con base en su estructura. El desarrollo de los generadores es una tarea mucho más fácil, y está más o menos claro que algoritmos son necesarios en estos sistemas. Al contrario, en el desarrollo de los analizadores sintácticos, todavía es un problema, especialmente para los idiomas que no tienen un orden de palabras fijos, como en el español, por ello las teorías basadas en inglés no son fáciles da adoptar al español.

Semántica

Semántica [6] (proviene del griego *semantikos***,** "lo que tiene significado"), es el estudio del significado de los signos lingüísticos (palabras, expresiones y oraciones).

Para ello se tienen que estudiar qué significa para los hablantes, cómo los designan, y también cómo los interpretan los oyentes.

Todo signo tiene dos vertientes: el significante o parte material del mismo y el significado o imagen que mentalmente genera el significante. Además, hay que distinguir otro elemento que es el referente o elemento real, al que se refieren tanto significado como significante. No es lo mismo la palabra que designa un referente que el referente mismo.

El significado o imagen mental está compuesto por una serie de rasgos que todos los hablantes de una lengua asocian de una manera general a un significante. No obstante, este significado tiene dos componentes:

- **Denotación.** Son los rasgos objetivos. Es el significado concreto de una palabra fuera de contexto. Constituyen el núcleo semántico fundamental. Comunes para todos los hablantes. Es el significado que se puede encontrar en el diccionario.
- **Connotación**. Son los rasgos subjetivos. Son las significaciones que lleva añadidas una palabra de manera subjetiva. De modo que dependiendo de los hablantes, época o lugar no cobre un significado distinto. Como por ejemplo, *goma* en el idioma castellano (plástico) y en el español de Sudamérica (neumático).

La semántica estudia las diferentes relaciones existentes entre un signo y todos los demás dentro de un contexto llamado léxico, que es el que constituye un sistema que permite a los hablantes la interpretación, conocimiento, adquisición y uso de ese léxico.

- **Relaciones entre significantes: la homonimia**

Se dice que dos palabras son homónimas si su significante es el mismo; es decir, están compuestas por los mismos fonemas, o su realización fonética coincide. No se trata de relación entre significados. La relación homonímica más habitual se produce entre palabras de distinta categoría gramatical como se puede ver en la siguiente tabla.

Homonimia (I)	
Cojo	Adjetivo. Persona o animal que cojea o le falta un pie o pierna.
Cojo	Primera persona del singular del presente indicativo del verbo coger.

Tabla 1. Homonimia (I)

Pero también se produce en palabras de la misma categoría**.** Se da en aquellos casos en que el significado de las palabras no tiene ninguna relación, porque proceden de étimos distintos.

Homonimia (II)	
Granada	Fruta
Granada	Proyectil

Tabla 2. Homonimia (II)

Dentro del concepto general de homonimia, se pueden distinguir varios casos mostrados a continuación.

Palabras homógrafas: Tienen las mismas grafías y los mismos sonidos.

Palabras homógrafas	
Asta	Cuerno
Asta	Palo de la bandera

Tabla 3. Palabras homógrafas

Palabras homófonas: Tienen los mismos sonidos, pero distinta grafía.

Palabras homófonas	
Asta	Palo de la bandera
Hasta	Preposición.

Tabla 4. Palabras homófonas.

Todas ellas son homónimas. Las dos primeras son homógrafas, y las dos últimas son homófonas, entre sí, y respecto a las anteriores.

- **Separación de relaciones entre significado y significante: monosemia, polisemia y sinonimia**
  - **Monosemia**

Es la relación habitual que existe entre el significado y el significante en una palabra. A un significante se corresponde un solo significado.

Por ejemplo, la palabra *bolígrafo* expresa un referente que sólo puede ser evocado mediante ese significante.

- **Polisemia**

Una palabra es polisémica cuando se puede expresar con ella varios significados. O, dicho de otra forma: un significante puede tener varios significados.

La polisemia se distingue de la homonimia en que se trata de una relación entre los dos planos del signo lingüístico: los diferentes significados de una palabra tienen, o han tenido, un origen común.

Polisemia	
Vela	Cilindro de cera o pieza de lona
Cuarto	Habitación o número fraccionario

Tabla 5.Polisemia

La polisemia es uno de los mecanismos más eficaces de economía lingüística, pues permite expresar varios significados con un único significante.

- **Sinonimia**

Dos o más palabras son sinónimas si tienen el mismo significado. Es decir, la sinonimia consiste en la igualdad de significado, cuando existen diferentes significantes.

Se pueden distinguir diversas formas en que puede presentarse la sinonimia:

- **Sinonimia conceptual:** Los significados denotativos son plenamente coincidentes. Ej.: *mujer =esposa*
- **Sinonimia connotativa:** Puede, en ocasiones, no haber coincidencia denotativa; sin embargo, esto no impediría que se consideren sinónimos por los valores connotativos que encierran. Ej.: *espabilado =avispado.*
- **Sinonimia contextual:** En determinados contextos, se pueden establecer ciertas sinonimias que serían impensables en otros. Ej.: *pesado = indigesto*, el cocido es pesado.

➢ **Relaciones entre significados: el campo semántico**

- **Hiperonimia e hiponimia**

Se llama hiperónimo [7] a la palabra cuyo significado abarca al de otras, que se conocen como hipónimos [8]. Los hipónimos a los que se refiere una palabra son, entre sí, cohipónimos. Se pueden distinguir las relaciones siguientes ilustradas con ejemplos en tablas.

- **Relaciones de inclusión:** Un conjunto de palabras puede estar englobado dentro de otra palabra que las incluya a todas.

Hiperónimo	Hipónimos	
Animal	Tigre	Hipónimos
	León	
	Lince	

Tabla 6. Hipónimos

- **Relaciones lineales.** En otros casos, se establecen relaciones de sucesión. Así sucede, con los nombres de los meses o los días: *enero, febrero,..., diciembre; lunes, martes,..., domingo*.

Hiperónimo	Hipónimos	
Meses	Enero	Cohipónimos
	Febrero	
	Marzo	

Tabla 7. Cohipónimos

- **Valores expresivos del significado**

El significado puede convertirse en un elemento de máxima efectividad expresiva. Si se tienen en cuenta los elementos de la comunicación, la situación comunicativa aclarará el significado de muchas palabras. Pero a veces, el contexto referencial hará que surjan significados nuevos, que antes no estaban presentes.

Hay que tener en cuenta que toda palabra tiene un significado denotativo y un significado connotativo. Las connotaciones pueden ser positivas o negativas, siempre dependiendo del hablante que las considere.

Palabras tabú son aquellas que no se pronuncian, porque tienen una carga connotativa despectiva. Se sustituyen por otras palabras que designan la misma realidad, pero sin esas connotaciones peyorativas. Son los denominados eufemismos (del griego: palabra bien sonante).

En ciertas ocasiones de la historia, el uso de ciertas palabras puede herir la sensibilidad de la sociedad. Por ejemplo, en estos tiempos de crisis en los que nos encontramos, los medios de comunicación prefieren decir que una empresa ha hecho una *regulación de empleo*, a decir, un *despido masivo* para evitar el pánico colectivo.

Al igual que existen eufemismos, también se emplean disfemismos**.** Cuando la palabra tabú se sustituye por otra, pero de carácter humorístico. Ej: En lugar de *morir,* se utiliza el disfemismo *estirar la pata.*

Pragmática

La pragmática [9] es el estudio del modo en que el contexto influye en la interpretación del significado. El contexto debe entenderse como situación, ya que puede incluir cualquier aspecto extralingüístico.

Incluye en sus análisis los factores sociales, psicológicos, culturales, literarios, que determina la estructura de la comunicación verbal y sus consecuencias. En este nivel se relacionan la semántica y la sintaxis: la semántica hace abstracción de los usuarios y la sintaxis expresa la relación entre los signos sin tener en cuenta a los usuarios; sintetizando todo el proceso en el estudio del qué se dice y lo que literalmente se quiere decir.

Es fundamental analizar también las huellas que emisor y receptor dejan en el texto. Así, por ejemplo, la presencia de un YO que se dirige a un TÚ puede imprimir una cierta fuerza persuasoria al mensaje, al introducirse, consciente o inconscientemente, el autor en el texto en un intento de modificar la conducta de la persona que recibe el mensaje.

La pragmática busca analizar los cambios que se presentan en determinados contextos, porque estos contextos permean las palabras, los gestos y el mensaje en general de hablantes, y la o las interpretaciones hechas por los oyentes.

2.1.3 Problemas de la lingüística en el procesamiento del lenguaje natural

Los distintos niveles vistos anteriormente causan diversos problemas a los que se enfrenta el procesamiento del lenguaje natural. Algunos de los problemas son de difícil resolución y otros son irresolubles.

El problema que se da siempre es la ambigüedad, bien sea por su manera de escribirse (*banco,* entidad financiera; *banco,* asiento en el parque; *banco,* acumulación de arena en un rio) o bien por su manera de pronunciarse (*vaca*, animal; *baca*, sujeción del coche), como ya se vio en los apartados anteriores. La ambigüedad a su vez acarrea problemas que se van arrastrando a lo largo del análisis, tanto del morfológico como del sintáctico posteriormente. Con lo que es el principal problema al que se ha de enfrentar cualquier sistema de procesamiento de lenguaje natural.

En la siguiente tabla se muestra, a modo de resumen, los problemas que se generan por niveles de menor a mayor.

Disciplina	Problema
Fonología	Afecta a los programas que utilicen reconocedor de voz Problemas con las pautas de voz de los usuarios Ej: Vaca / Baca Hora / Ora
Morfología	Confusiones a la hora de realizar el análisis morfológico en palabras que se escriben igual (homógrafas) Ej: "La casa" →verbo casar → edificio
Sintaxis	Concordancia de género y número
Semántica	La homonimia y la polisemia
Pragmática	Desconocimiento del contexto Problemas con la persona. Ej: "¿Tienes hora?"

Tabla 8. Problemas de cada disciplina

2.2 Procesamiento del Lenguaje Natural (PLN)

2.2.1 Introducción

El procesamiento del lenguaje natural o Natural Language Processing(PLN o NLP) [10] es una disciplina de la ingeniería lingüística que se encarga de estudiar de qué manera se puede mejorar el lenguaje natural con un lenguaje que entienda un ordenador, a través de algún dispositivo que interaccione con el usuario ya sea mediante texto o voz.

Por lo tanto el objetivo del PLN es procesar, traducir e interpretar de forma automática el lenguaje de los humanos para que, a través de una serie de algoritmos, las máquinas sean capaces de extraer la información correctamente. Es decir, es hacer una serie de sistemas y programas que sean capaces de recuperar la información que un usuario humano introduce en su propio idioma, (inglés, francés, español, etc.).

El procesamiento del lenguaje natural presenta múltiples aplicaciones:

- Corrección de textos
- Traducción automática
- Recuperación de la información
- Extracción de información y resúmenes
- Búsqueda de documentos
- Sistemas Inteligentes para la educación y el entrenamiento

La **corrección de textos** permite la detección y corrección de errores ortográficos y gramaticales. Para detectar este tipo de errores, la computadora necesita entender en cierto grado el sentido del texto. Los correctores de gramática detectan las estructuras incorrectas en las oraciones aunque todas las palabras en la oración estén bien escritas en el lenguaje en cuestión. El problema de detectar los errores de este tipo es complejo debido a la existencia de gran variedad de estructuras permitidas.

Para describir las estructuras de las oraciones en el idioma, se usan las llamadas gramáticas formales, o sea conjuntos de reglas de combinación de palabras y su orden relativo en las oraciones.

La **traducción automática** se refiere a la traducción correcta de un lenguaje a otro, tomando en cuenta lo que se quiere expresar en cada oración.

En el campo de la **recuperación de la información** se han desarrollado sistemas que permiten obtener información sobre estadísticas deportivas, información turística, geografía etc. En lugar de buscar los documentos para encontrar en ellos la respuesta a su pregunta, el usuario podría hacer su pregunta a la computadora: *¿Cómo se llama el presidente de Francia?, ¿Cuáles son los centros más avanzados en Procesamiento del Lenguaje Natural?,* y otras.

Por otra parte, se han desarrollado sistemas con la capacidad de **crear resúmenes** de documentos a partir de los datos suministrados. Estos sistemas son capaces de realizar un análisis detallado del contenido del texto y elaborar un resumen.

También se han desarrollado sistemas inteligentes que **permiten modelar el comportamiento** del estudiante, reconocer y procesar sus errores, desarrollar habilidades en la resolución de problemas y otras actividades del proceso enseñanza y aprendizaje. En ellos el Procesamiento del Lenguaje Natural juega un papel de relevante importancia en la creación y desarrollo de interfaces amigables**.**

2.2.2 Arquitectura típica de un sistema de Procesamiento del Lenguaje Natural

Uno de los elementos fundamentales en el diseño de un sistema PLN es sin lugar a dudas la determinación de la arquitectura del sistema, es decir, como se introducen los datos a la computadora y como ella interpreta y analiza las oraciones que le sean proporcionadas. A continuación se muestra un esquema del análisis léxico/sintáctico por computadora. El sistema consiste de:

- El usuario le expresa (de alguna forma) a la computadora que tipo de procesamiento desea hacer.
- La computadora analiza las oraciones proporcionadas, en el sentido morfológico y sintáctico;
- Luego, se analizan las oraciones semánticamente, es decir se determina el significado de cada oración;
- Se realiza el análisis pragmático del texto. Así, se obtiene una expresión final.

Se ejecuta la expresión final y se entrega al usuario para su consideración.

A continuación se muestra de una forma gráfica los pasos que se realizan de forma típica a la hora de implementar un PLN.

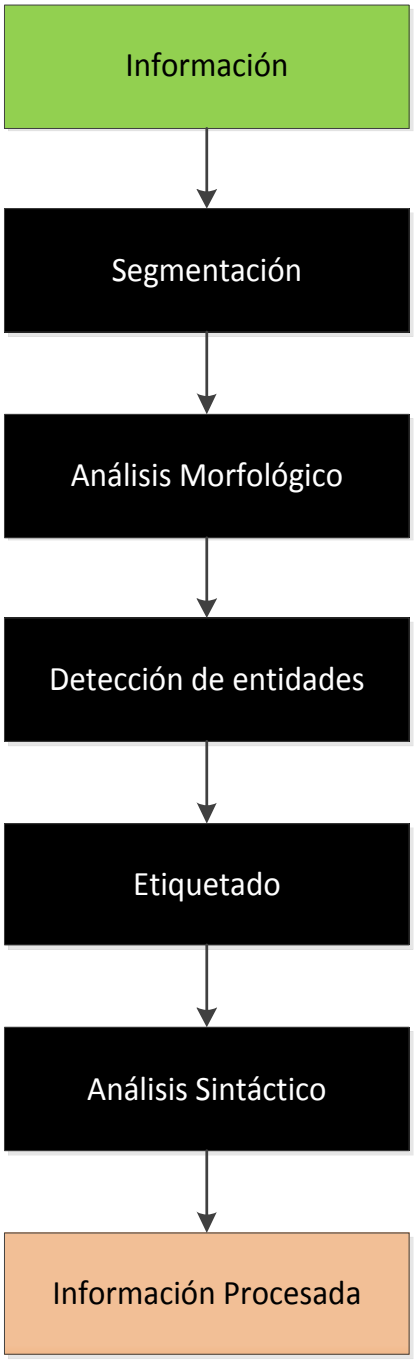


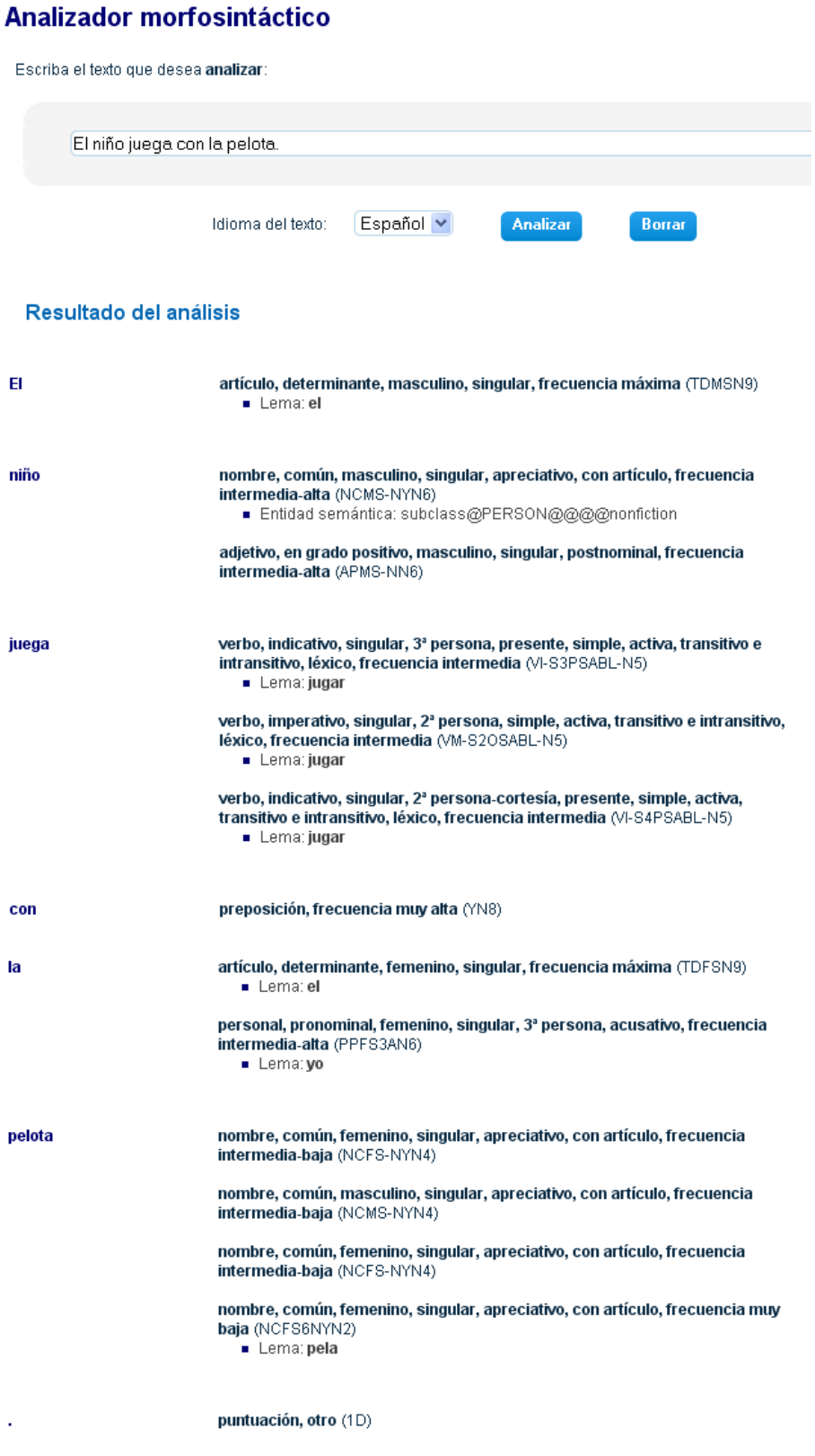
Imagen 1. Esquema de procesado típico PNL

2.2.3 Segmentación

Principalmente, la segmentación consiste en dividir el texto por frases, y estas frases a su vez en palabras. De este modo, una vez se tienen las palabras por separado, puedan ser categorizadas y se interprete el significado que aportan al conjunto del que son parte.

2.2.4 Análisis Morfológico

El análisis morfológico consiste en determinar la forma, clase o categoría gramatical de cada palabra de una oración. Un ejemplo es el mostrado en la figura siguiente (utilizando la herramienta STILUS [11]).



Analizador morfosintáctico

Escriba el texto que desea **analizar**:

El niño juega con la pelota.

Idioma del texto: Español Analizar Borrar

Resultado del análisis

El — **artículo, determinante, masculino, singular, frecuencia máxima** (TDMSN9)
- Lema: **el**

niño — **nombre, común, masculino, singular, apreciativo, con artículo, frecuencia intermedia-alta** (NCMS-NYN6)
- Entidad semántica: subclass@PERSON@@@@nonfiction

adjetivo, en grado positivo, masculino, singular, postnominal, frecuencia intermedia-alta (APMS-NN6)

juega — **verbo, indicativo, singular, 3ª persona, presente, simple, activa, transitivo e intransitivo, léxico, frecuencia intermedia** (VI-S3PSABL-N5)
- Lema: **jugar**

verbo, imperativo, singular, 2ª persona, simple, activa, transitivo e intransitivo, léxico, frecuencia intermedia (VM-S2OSABL-N5)
- Lema: **jugar**

verbo, indicativo, singular, 2ª persona-cortesía, presente, simple, activa, transitivo e intransitivo, léxico, frecuencia intermedia (VI-S4PSABL-N5)
- Lema: **jugar**

con — **preposición, frecuencia muy alta** (YN8)

la — **artículo, determinante, femenino, singular, frecuencia máxima** (TDFSN9)
- Lema: **el**

personal, pronominal, femenino, singular, 3ª persona, acusativo, frecuencia intermedia-alta (PPFS3AN6)
- Lema: **yo**

pelota — **nombre, común, femenino, singular, apreciativo, con artículo, frecuencia intermedia-baja** (NCFS-NYN4)

nombre, común, masculino, singular, apreciativo, con artículo, frecuencia intermedia-baja (NCMS-NYN4)

nombre, común, femenino, singular, apreciativo, con artículo, frecuencia intermedia-baja (NCFS-NYN4)

nombre, común, femenino, singular, apreciativo, con artículo, frecuencia muy baja (NCFS6NYN2)
- Lema: **pela**

. — **puntuación, otro** (1D)

Imagen 2. Ejemplo de análisis morfosintáctico con la herramienta STILUS

Como se aprecia en la ilustración, realizar el análisis morfológico realmente consiste en dar una etiqueta o marca a cada palabra, bien sea un código que interpretará posteriormente el programa o bien sean dándole los rasgos completos.

A la hora de realizar este etiquetado, surge un problema que después jugará un papel muy importante para realizar correctamente los análisis o no, la desambigüación.

En el caso de STILUS, el etiquetado consiste en asignar una marca (etiqueta) formada por cifras y letras en las que se indica la categoría gramatical, género, número, persona, tiempo verbal, modo…

Esto ayuda a elegir las palabras que se van a buscar ya que es muy común dar más importancia a un nombre que a una preposición.

Para el caso particular de este proyecto se optó por la opción de dar un código a cada palabra, más adelante, en el apartado del diseño se especificará que tipo de palabra se corresponde con su código.

2.2.5 Detección de Entidades

Algunos sistemas y herramientas del procesamiento del lenguaje natural son capaces de detectar entidades tipo fechas, ciudades, personas, etc. Estas detecciones permiten encontrar multipalabras que se refieren a un mismo objeto pero que se encuentra fraccionado, como las fechas como se aprecia en el ejemplo, o los nombres propios. Ejemplo: "*Rafael Nadal*" se debe referir a la misma persona, no "*Rafael*" por una parte al cantante y "*Nadal*" al tenista o jugador de fútbol.

En la siguiente imagen se muestra un ejemplo de detección de entidades que hace la herramienta STILUS.

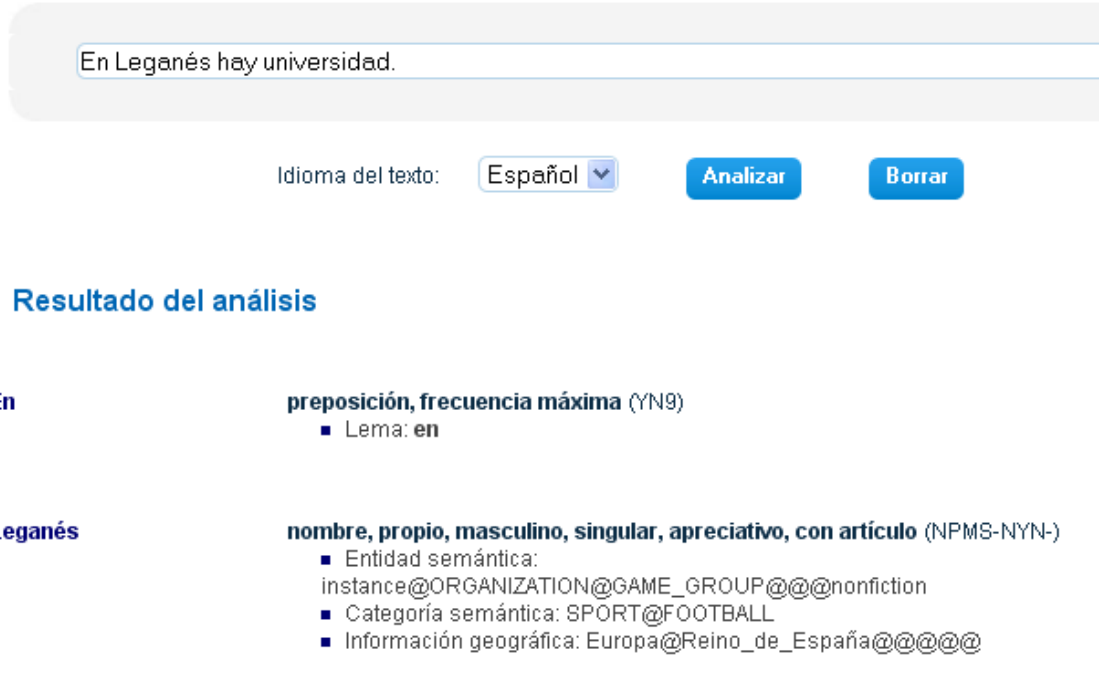


Imagen 3. Ejemplo de detección de entidades con la herramienta STILUS.

En este proyecto se usa la detección de entidades de tipo nombres propios, los cuales deberán estar escritos correctamente. En el apartado de diseño se detallarán las principales características de detección de entidades desarrolladas en este proyecto.

2.2.6 Etiquetado

Este etiquetado se basa en lematizar cada una de las palabras en las que se ha fraccionado el texto. Lematizar consiste en la reducción de las diferentes formas flexivas de una palabra a la forma canónica, su lema o representante de toda la familia flexiva.

Por convenio esta reducción consiste en reagrupar las distintas inflexiones de un verbo en el infinitivo; el singular y el plural de un sustantivo en el singular; el masculino y el femenino de un adjetivo en el masculino. Con esto se consigue identificar familias de palabras para considerarlas como una sola. Es un dato muy importante para la búsqueda de información ya que se consigue hacer independiente la búsqueda de tiempos verbales y otros.

2.2.7 Análisis Sintáctico

De todos los niveles de análisis, la sintaxis ha sido durante mucho tiempo y aún sigue siendo el nivel al que la lingüística le ha prestado mayor atención. Esta casi exclusiva atención se justifica por dos razones principales en cuanto al tratamiento automático del lenguaje natural:

- El procesamiento semántico funciona sobre los constituyentes de la oración. Si no existe un paso de análisis sintáctico, el sistema semántico debe identificar sus propios constituyentes. Por otro lado, si se realiza un análisis sintáctico, se restringe enormemente el número de constituyentes a considerar por el semántico, mucho más complejo y menos fiable. El análisis sintáctico es mucho menos costoso computacionalmente hablando que el análisis semántico (que requiere inferencias importantes). Por tanto, la existencia de un análisis sintáctico conlleva un considerable ahorro de recursos y una disminución de la complejidad del sistema.
- Aunque frecuentemente se puede extraer el significado de una oración sin usar hechos gramaticales, no siempre es posible hacerlo. La sintaxis contempla dos modos diferentes, pero no por ello opuestos, de análisis. El primero es el análisis

de constituyentes o análisis de estructura de frase: la estructuración de las oraciones en sus partes constituyentes y la categorización de estas partes como nominales, verbales, adjetivales, etc. El segundo es el análisis de las relaciones o funciones gramaticales: la asignación de relacionales gramaticales tales como sujeto, objeto, etc.

2.2.8 Análisis Semántico

En muchas aplicaciones del PLN los objetivos del análisis apuntan hacia el procesamiento del significado. En los últimos años las técnicas de procesamiento sintáctico han experimentado avances significativos, resolviendo los problemas fundamentales, sin embargo, las técnicas de representación del significado no han obtenido los resultados deseados, y numerosas cuestiones continúan sin encontrar soluciones satisfactorias.

Definir qué es el significado no es una tarea sencilla, y puede dar lugar a diversas interpretaciones. A efectos prácticos es necesaria una buena modularidad para facilitar el procesamiento, de tal manera que sea posible distinguir entre significado independiente o dependiente del contexto. El significado independiente del contexto, tratado por la semántica, hace referencia al significado que las palabras tienen por sí mismas sin considerar el significado adquirido según las circunstancias en las que se está usando dicha palabra, es decir, se hace referencia a las condiciones de verdad de la frase, ignorando la influencia del contexto o las intenciones del hablante. Por su parte, el significado dependiente del contexto, estudiado por la pragmática, se refiere al componente significativo de una frase asociado a las circunstancias en que ésta se utiliza.

Atendiendo al desarrollo en el proceso de interpretación semántica, es posible optar entre múltiples pautas para su organización, explicadas a continuación.

En referencia a la estructura semántica que se va a generar, puede interesarnos que exista una simetría respecto a la estructura sintáctica, de tal manera que se generará una estructura arbórea para el análisis semántico que tendrá las mismas características que el árbol sintáctico, o por el contrario que no se dé tal correspondencia entre ellas, caso en el que se realizarán varias transformaciones sobre la estructura utilizada en la sintaxis, generándose la representación semántica sobre dichas transformaciones.

Cada una de las dos opciones anteriores puede implementarse de forma secuencial (en primer lugar se realiza el análisis sintáctico y, una vez finalizado éste, se pasa al análisis semántico) o paralela (se puede iniciar el análisis semántico de cada constituyente cuando éste ha sido tratado por el analizador sintáctico).

Finalmente, en combinación con cada una de las opciones anteriores, podemos escoger un modelo en el que exista una correspondencia entre reglas sintácticas y semánticas o, contrariamente, podemos optar por un modelo que no cumpla tal requisito. En caso afirmativo, para cada regla sintáctica existirá una regla semántica correspondiente.

El significado es representado por formalismos conocidos por el nombre de *knowledge representation*. El léxico proporciona el componente semántico de cada palabra en un formalismo concreto, y el analizador semántico lo procesa para obtener una representación del significado de la frase.

2.3 XML

Las siglas en inglés XML[12] provienen de "*e**X**tensible **M**arkup **L**anguage*" lo que traducido al español es un lenguaje de marcas extensible. XML fue creado por el *World Wide Wev Consortium*[13] (W3C) como un metalenguaje extensible de etiquetas.

Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. De esta forma XML no es en realidad un lenguaje específico, sino una forma de definir lenguajes para distintas necesidades, es por ello que se le llama metalenguaje. Algunos de estos lenguajes son XHTML, SVG, por ejemplo.

Se podría considerar que XML es un lenguaje de metamarcado que ofrece un formato para la descripción de datos estructurados. Esto facilita unas declaraciones de contenido más precisas y unos resultados de búsquedas más significativos en varias plataformas, con esto se propone como un estándar para el intercambio de información estructurada. XML se puede usar en editores de texto, bases de datos, hojas de cálculo…

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante

en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

2.3.1 Ventajas de XML

XML presenta múltiples ventajas, a continuación se muestran las más relevantes:

- Es extensible. Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Por lo tanto se mejora la compatibilidad entre aplicaciones. Se pueden comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos.
- Existe una transformación de datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.
- Los autores y proveedores pueden diseñar sus propios tipos de documentos usando XML.
- La información contenida puede ser más 'rica' y fácil de usar, porque las habilidades hipertextuales de XML son amplias.
- XML puede dar más y mejores facilidades para la representación en los visualizadores.
- La información es más accesible y reutilizable, porque la flexibilidad de las etiquetas de XML pueden utilizarse sin tener que amoldarse a reglas específicas.
- Los archivos XML válidos son válidos también en SGML, luego pueden utilizarse también fuera de la Web, en un entorno SGML (una vez la especificación sea estable y el software SGML la adopte).
- Elimina muchas de las complejidades de SGML, en favor de la flexibilidad del modelo, con lo que la escritura de programas para manejar XML será más sencilla que haciendo el mismo trabajo en SGML.

- Es muy sencillo de usar, además cuenta con múltiples tutoriales en Internet, como puede ser el “w3cshools” [14].

2.3.2 Estructura de un documento XML

Aunque a primera vista, un documento XML puede parecer similar a HTML, hay una diferencia principal. Un documento XML contiene datos que se autodefinen, exclusivamente. Un documento HTML contiene datos mal definidos, mezclados con elementos de formato. En XML se separa el contenido de la presentación de forma total.

Una forma de entender rápidamente la estructura de un documento XML es viendo un pequeño ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
<remite>
<nombre>Alfredo Reino</nombre>
<email>alf@ibium.com</email>
</remite>
<destinatario>
<nombre>Bill Clinton</nombre>
<email>president@whitehouse.gov</email>
</destinatario>
<asunto>Hola Bill</asunto>
<texto>
<parrafo>¿Hola qué tal? Hace <enfasis>mucho</enfasis> que no escribes. A ver si llamas y quedamos para tomar algo.</parrafo>
</texto>
</mensaje>
```

Este mismo documento puede ser visto de forma gráfica, para comprender mejor la estructura de un documento XML.

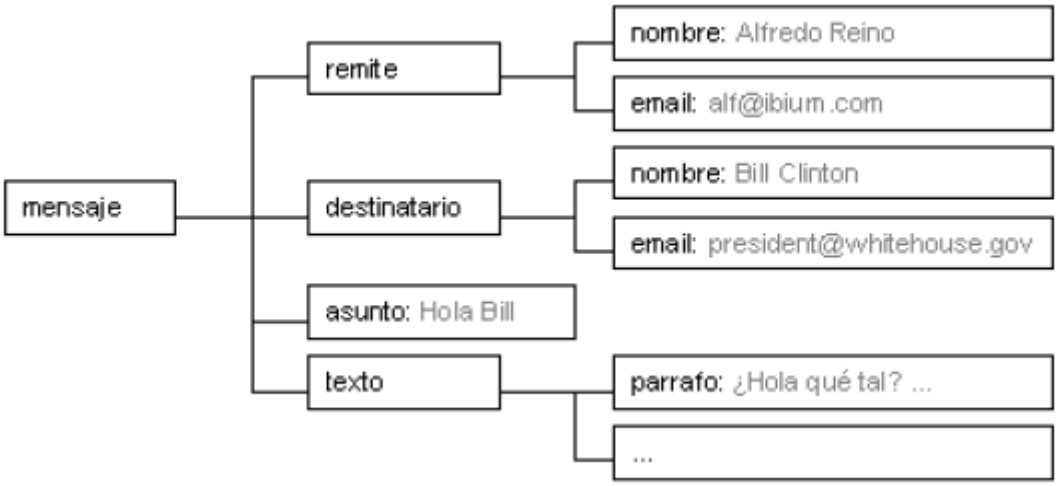


Imagen 4. Ejemplo árbol XML.

3 Diseño del sistema

3.1 Introducción

El sistema consiste en la obtención de un lenguaje formalizado con información estructurada a partir de información no estructurada usando un procesador de lenguaje natural.

A continuación se muestra el diseño general del sistema.

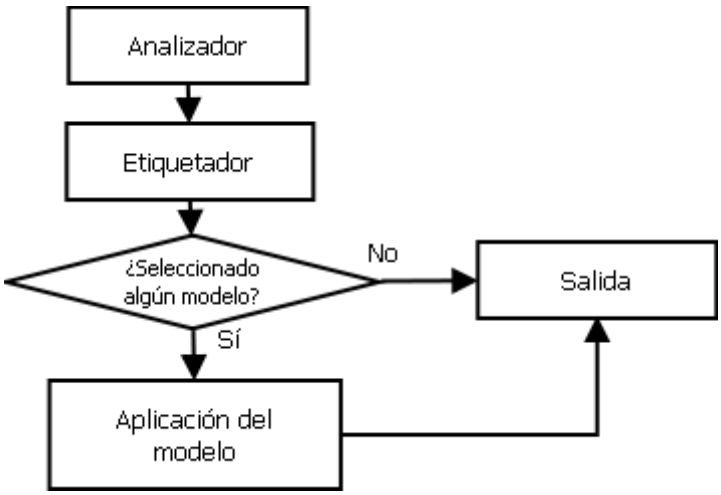


Imagen 5. Esquema general del sistema

En primer lugar se van a describir las funciones que realiza el bloque de análisis de lenguaje natural así como los estados por los que va pasando el texto inicial.

El siguiente bloque es el etiquetador y es donde el sistema trata de identificar términos semánticos y etiquetarlos.

Una vez realizado el etiquetado del lenguaje se pasa a la formalización de la salida en función del dominio que se haya seleccionado, este proceso se realiza mediante unos modelos. Estos modelos son un conjunto de términos y reglas que permiten caracterizar la información, separando lo que realmente es información de lo que son palabras vacías.

Por último se representará la información de una forma más fácil de entender para la máquina. La salida será en texto o en XML.

Se trata de un procesador de lenguaje natural en el cual no se tiene una base de datos en la que incluye todos los términos de un diccionario, simplemente se basa en listas de palabras para categorizar el resto de elementos. Habitualmente en los procesadores de lenguaje natural se utiliza una base de datos muy grande en la que se incluyen todos los términos de una lengua, aquí no se usa para mejorar el rendimiento y únicamente el sistema se basa en listas de palabras.

El diseño del sistema parte con la premisa de que los datos de entrada son correctos y que están correctamente escritos, tanto ortográficamente como semánticamente. No obstante, como se trata de listas de palabras se pueden añadir términos para disminuir el número de errores por mala interpretación del sistema (principalmente estos errores suelen ser ortográficos).

3.2 Analizador

Este subsistema es el encargado de obtener la información que le pasa el usuario y transformarla en una estructura de datos que sea más sencilla de analizar y procesar.

A continuación se muestra el esquema de la estructura de este subsistema.

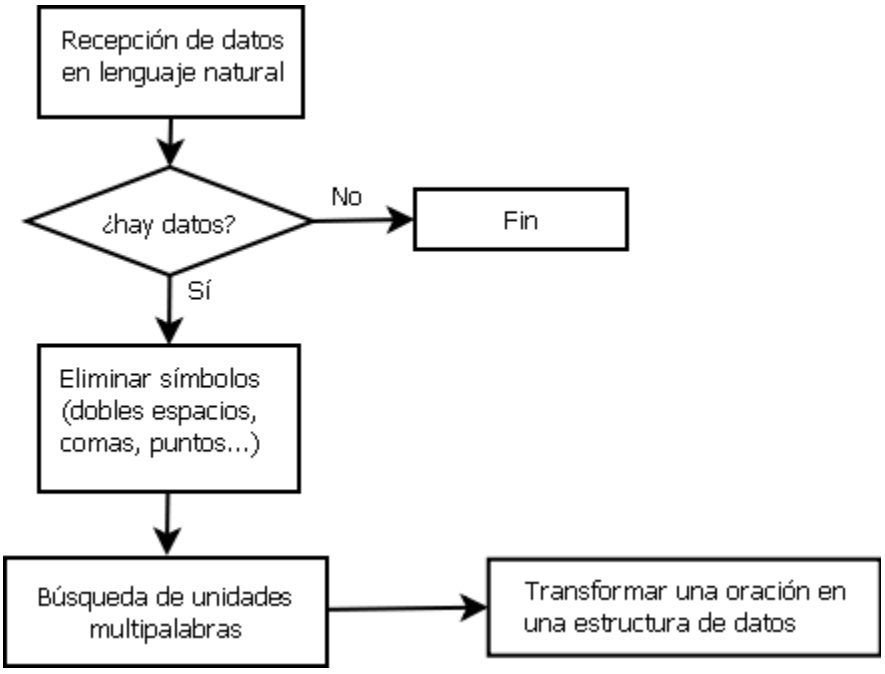


Imagen 6. Esquema módulo analizador.

La primera acción que se realiza es la extracción de información de la consulta del usuario. Típicamente esta consulta está pensada para que se reciba mediante un formulario de una página web y el sistema procese estos datos del formulario. Posteriormente se procederá a la comprobación de que haya datos, es decir, que la consulta no esté vacía.

El siguiente punto que se realiza es la eliminación de elementos que pueden molestar en los siguientes subsistemas. Estos elementos pueden ser dobles espacios, exclamaciones, interrogaciones, dos puntos, símbolos, etc.

Para evitar problemas con el reconocimiento de nombres se ha optado por cambiar siempre la primera letra de la oración a minúscula, con esto se trata de evitar que el sistema reconozca como nombres propios palabras comunes.

Para la búsqueda de multipalabras se requiere una lista de palabras para comparar término a término si corresponden dichas palabras a una multipalabra. La forma más sencilla de proceder es que las palabras que formen un conjunto como multipalabra se separen con guión bajo.

Una vez que se tiene la lista de palabras se compara con el texto inicial en busca de alguna coincidencia, en caso de encontrar alguna se pone el término como en la lista de palabras. De esta forma, se evitará que posteriormente, cuando el sistema transforme el texto de entrada a una estructura de datos, se coja cada término de la multipalabra por separado.

El siguiente proceso que se realiza es el de la transformación de la oración que ha introducido el usuario a una estructura de datos determinada, la cual facilitará el proceso en los siguientes subsistemas.

A continuación se muestra un ejemplo de cómo quedaría si no actuase la lista de multipalabras.

Ejemplo :

Posición en la estructura de datos	Texto original: ¿Cuánto se tarda a Valencia?
0	Cuánto
1	se
2	tarda
3	a
4	Valencia

El sistema transforma los datos de tal forma que cada término se corresponde a una posición en la estructura de datos, para conseguirlo se buscan los espacios entre los diferentes términos.

Como se trata de una multipalabra, el sistema la ha encontrado y los espacios entre las palabras los ha sustituido por guiones bajos, entonces, al no tener espacios entre las palabras el sistema sólo lo cogerá en una única posición. Una vez que los datos han sido introducidos en la estructura de datos se procede a la sustitución de los guiones bajos por espacios, para dejarlos de la manera inicial.

Debajo se muestra un ejemplo de la estructura real que quedaría aplicándose la lista de multipalabras.

Ejemplo :

1. cuánto se tarda

Son tres palabras que su significado conjunto expresa cantidad de tiempo o simplemente tiempo. En el ejemplo anterior se puede ver que ocupan 3 posiciones en la estructura de datos, a partir de encontrar estas palabras entre las multipalabras se situarán en una única posición.

Posición en la estructura de datos	Texto original: ¿Cuánto se tarda a Valencia?
0	Cuánto se tarda
1	a
2	Valencia

Anteriormente se ha expresado que el sistema no corregía los errores gramaticales, pero debido a que era una fuente alta de errores se optó por poner un corrector de meses, ya que estos siempre deben ir en minúsculas. Se trata de una operación sencilla y garantiza el funcionamiento del sistema correctamente.

3.3 Etiquetado

El principal cometido de este subsistema es el de extraer cierto tipo de información de las palabras introducidas por el usuario. El subsistema tratará de etiquetar las palabras que reconozca para que una vez etiquetadas sea más sencillo discriminar qué palabras son válidas y qué palabras no aportan información útil.

Los tipos de palabras se pueden etiquetar en el sistema como: nombres propios, preposiciones, determinantes, números, fechas, pronombres interrogativos, palabras vacías, datos válidos y nombres comunes (solamente detectados si les preceden un artículo determinado o indeterminado).

Un esquema simple de este modulo es el siguiente:

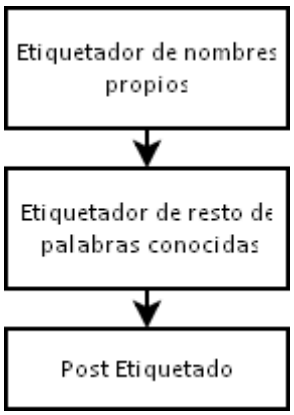


Imagen 7. Esquema módulo extracción de información.

A continuación se explicará más detalladamente lo que realiza cada bloque.

3.3.1 Reconocimiento de nombres propios

Esta parte del sistema se encarga de identificar qué palabra o palabras se pueden categorizar como nombres propios, adjuntando una etiqueta de nombre propio. En caso de que el nombre propio cuente con más de una palabra, el sistema reagrupará en una única posición todas las palabras para tratarlas de forma conjunta como nombre propio.

El sistema cuenta con dos formas de comprensión de nombres propios. La primera, más sencilla, es que si encuentra algo que venga entre comillas lo entiende como una unidad literal y lo pone en una única posición, tratándolo de forma similar que un nombre propio. Un ejemplo para lo que puede ser útil es para los títulos de los libros.

La segunda y más complicada es el reconocimiento de nombres propios a través de localizar la primera letra en mayúscula. Una vez localizada la primera letra hay que comprobar si las siguientes palabras pueden tener relación con el nombre, para ello se comprueba si la siguiente palabra: empieza por mayúscula, es la preposición “de”, es la formación “de la” o el artículo “la”. Como ejemplo de lo anterior expuesto, en orden de aparición: Francisco Fuentes, Zafra de Záncara, Madrigal de la Vera y Castilla la Mancha.

A continuación se muestra un diagrama de flujo para analizar el subsistema.

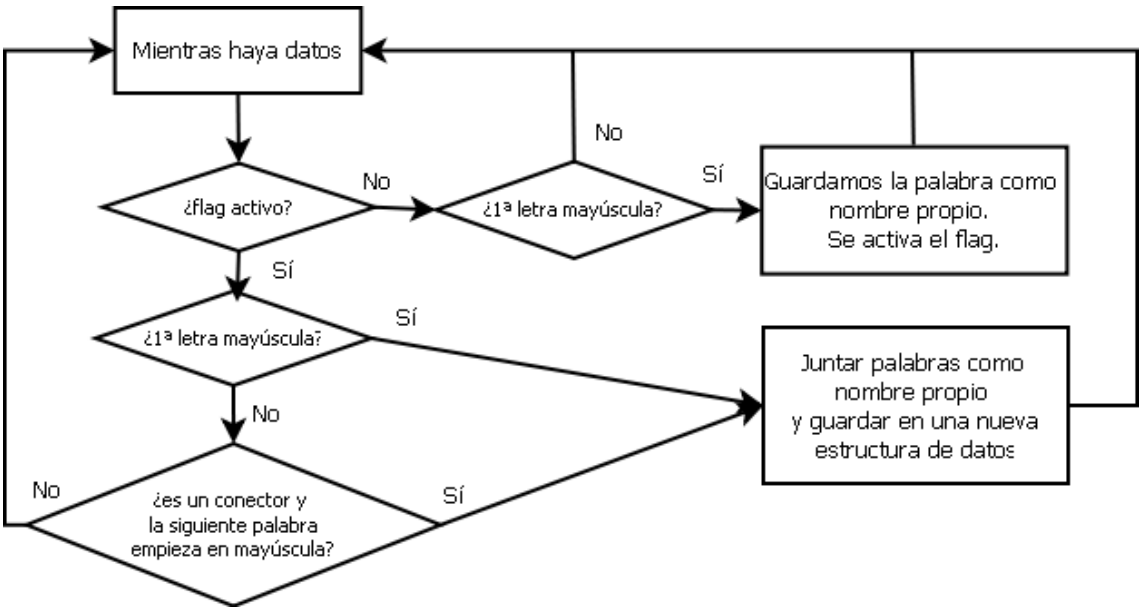


Imagen 8. Esquema reconocimiento nombre propio.

Los datos se reciben del anterior módulo, el analizador, por lo tanto aquí se tiene una estructura de datos para tratar. En primer lugar, como se dijo anteriormente, se procesan los datos que vengan entre comillas simples como si fuese un nombre propio.

Posteriormente se comprueba si un flag está activado. Este flag es una variable que indica si la anterior palabra empezaba con mayúscula. En caso negativo se comprueba que la palabra tiene la primera letra en mayúscula, si es así se pone el flag activo para la siguiente secuencia y el valor se guarda en una nueva estructura de datos, en este caso se guardaría el nombre propio. En caso contrario se vuelve al principio ya que se trata de una palabra toda ella en minúsculas.

Si el flag está activado hay varios casos. El primero que se da es que la siguiente palabra tenga su primera letra también en mayúscula, por lo tanto se unen formando una única entidad como nombre propio, en caso que haya más palabras de forma continuada con su primera letra en mayúscula se van juntando con las anteriores, ya que se comportaría de la misma manera que en el caso anterior.

El segundo caso es que no se encuentre una palabra con su primera letra en mayúscula, entonces se comprueba si es alguno de los conectores “de”, “de la”, “la”, “del”. En caso positivo se mira la siguiente palabra, si su primera letra es mayúscula se procede a guardarse

todo el conjunto de palabras en una única posición. De no ser positivo se vuelve al principio, desactivando el flag.

El proceso se repite mientras haya datos.

Ejemplo:

- *Zafra de Záncara*

1.- Encuentra "Zafra"

Primera letra en mayúsculas → Sí

*Se guarda **Zafra** como nombre propio*

2.- Encuentra "de"

Primera letra en mayúsculas → No

¿Es alguna de las palabras "de", "la" o "del"? → Sí

¿La siguiente palabra empieza por mayúscula? → Sí

*Se guarda **Zafra de Záncara** como nombre propio.*

3.- Fin

3.3.2 Reconocimiento de tipos de palabras

Este módulo es el más importante del sistema ya que debido a su funcionalidad cambia y categoriza las diferentes palabras que le llegan desde la estructura de datos, así se consigue una estructura de datos más compleja en la cual muchas palabras están correctamente etiquetadas con respecto a la función que realizan en el sistema.

Los siguientes símbolos se corresponden con los distintos tipos de palabras que se pueden categorizar:

- **NP** => Nombre Propio
- **D** => Fecha
- **I** => Palabra importante
- **Q** => Término de Interrogación

- **N** => Cifra
- **NC** => Nombre común
- **T** => Hora
- **A** => Artículo
- **P** => Preposición
- **U** => Innecesario
- **V** => resto de datos sin identificar (datos válidos)

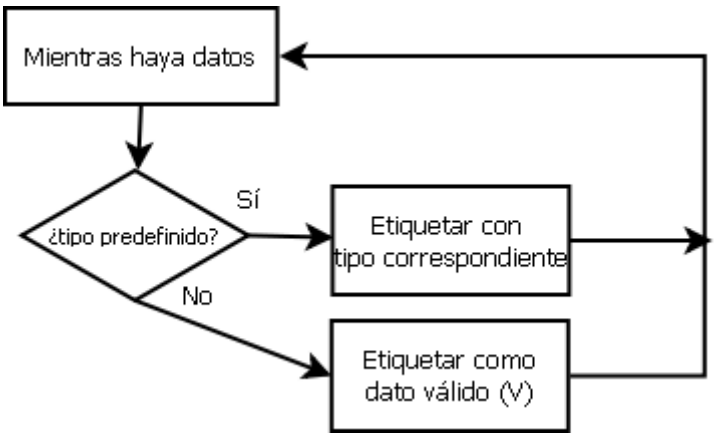


Imagen 9. Esquema reconocimiento de términos

La funcionalidad que tiene este bloque es la de etiquetar los datos estructurados en una nueva estructura de datos con más parámetros para categorizar ciertos tipos de términos.

La forma en la que se etiquetan los datos es mediante una búsqueda posición a posición, es decir, cada posición de la estructura de datos es comprobada con una serie de listas de palabras predefinidas, las palabras que se han predefinido para este proyecto se pueden encontrar en el Anexo. En caso de coincidir, se añade el tipo de dato que corresponda a la estructura de datos. En caso que no coincida con ninguna lista se categorizará como un dato válido.

Esta política se hace para no tener que disponer de una tremenda base de datos con todas las palabras etiquetadas. Así reducimos el tiempo de computación y etiquetamos las palabras que más suelen aparecer en las oraciones como pueden ser determinantes, preposiciones y palabras vacías (las que no aportan valor semántico).

Otras funcionalidades que hace este módulo son:

- simplificar el significado de multipalabras.
- transformar el nombre de un mes a su correspondiente número de mes.
- transformar y cambiar una fecha escrita a un formato estándar de fechas.

En el siguiente ejemplo se muestra como quedaría la nueva estructura de datos para algunas palabras.

Ejemplo:		
Dato = a	tipo = P	original = a
Dato = precio	tipo = Q	original = cuánto vale
Dato = viaje	tipo = V	original = viaje

A continuación se describe con más detalles los diferentes tipos que tiene el sistema.

Nombre Propio

La forma de etiquetar este tipo de entidades se ha detallado en el anterior apartado. A modo resumen, se categoriza como nombre propio a aquellas palabras que empiecen con mayúscula o las formaciones de primera letra en mayúscula más conectores más otra palabra que empiece en mayúscula. También se categorizan como nombres propios los términos que estén entre comillas.

Fechas

Se buscan elementos que se puedan identificar en su conjunto como una fecha, y posteriormente se transforman a un formato de fecha estándar.

Ejemplo

1. 13 de noviembre de 2011
2. 13/11/2011
3. 13-11-2011
4. Noviembre de 2011

Estos ejemplos excepto el último tendrían una salida como esta: 13-11-2011 con un atributo de fecha. El último caso al no estar especificado el día se pone el primer día del mes quedando 01-11-2011.

Palabras importantes

En este grupo se encuentran las conjunciones: “y”, “e”, “o”, “u” y las palabras “no”, “sin”, “ni”.

Se tratan estas palabras como importantes, ya que tienen la capacidad de cambiar el significado de la oración por eso merecen un trato especial.

Ejemplo

Vacaciones en Castellón y Valencia

Aquí el significado de “y” implica que el destino son las dos ciudades no una o la otra.

Término de Interrogación

Aquí se encuentran los términos que connotan un carácter interrogativo. Se encuentran en una lista para compararlos con los valores de la estructura de datos. Si alguno coincide se cambia por el valor que hay en la lista de interrogativos.

Ejemplo

Las palabras “cuánto tarda” se pueden reescribir por el término “tiempo”, quedando así más útil para la salida.

Cifra

Este tipo es muy fácil identificar ya que se trata de los números.

Nombre comunes

La única forma que se tiene de identificar sustantivos sin tener una base de datos es a través de los artículos, ya que tras ellos suele venir el sustantivo al que hace referencia. También decir que si viene un adjetivo y posteriormente el nombre habría un error ya que se categorizaría el adjetivo como nombre común.

Hora

Para identificar que se trata de un elemento de hora lo que se hace es mirar si los números llevan dos puntos entre ellos, en tal caso se califican como hora.

Otra forma de identificarlo es a través de los artículos “la” y “las” ya que hace referencia a las horas. Por ejemplo: “a las 5 de la tarde”. Aquí el sistema desambiguaría entre una cifra y una hora dependiendo de si precede algún artículo.

Artículo

En este apartado se encuentran los artículos, tanto los determinados como los indeterminados, en plural y en singular. Se trata de una lista donde se comprueba si coinciden con alguna posición de nuestra estructura de datos.

Preposición

Para determinar si se trata de una preposición, el sistema cuenta con una lista de preposiciones, exceptuando “sin”, porque debido a consideraciones de diseño no se incluye en esta lista ya que “sin” se considera dentro del grupo de palabras importantes.

Innecesario

Esta categoría cuenta con una lista de palabras que se denominan vacías, o en inglés “stopwords” qué es como más típicamente se les hace referencia. Se las denomina vacías

porque no aportan valor semántico a la oración generalmente y particularmente al sistema tampoco.

Una vez identificadas estas palabras mediante el método habitual de búsqueda se etiquetan con una U de (“unnecesary”). Estas palabras son candidatas para eliminarse del sistema.

Resto de datos sin identificar (datos válidos)

Todo lo que no ha sido clasificado en los anteriores campos se etiquetará de esta forma para su posterior tratamiento.

3.3.3 Post Etiquetado

Este bloque trata de mejorar la capacidad del etiquetador. Entre sus funciones destaca la traducción de los operadores matemáticos en forma de palabra a su correspondiente símbolo, traducción de algunas palabras que indican temporalidad a su fecha correspondiente. Por último, se realiza un cambio del formato de hora para adaptarlo al formato 24 horas.

La transformación de los operadores en sus correspondientes símbolos se debe hacer en este apartado y no en el de etiquetar. Esto es porque al tratarse de operadores matemáticos expresados en palabras puede haber ambigüedad y que no se trate en realidad de operadores matemáticos. Por lo tanto, se debe comprobar que haya números próximos. Si se hiciese al etiquetar, el diseño se complicaría debido a que cada vez que se encontrase una palabra del tipo operador, habría que buscar en toda la estructura de datos en busca de algún número.

Estos son los operadores matemáticos que se transforman y sus correspondientes símbolo: “menos, menor, antes, debajo” se transforman en “<” por otro lado “encima, mayor, más” se transforman en “>”.

Ejemplo

Vuelos a Canarias desde Madrid que cuesten menos de 100 Euros.

“Menos de” sería transformado a “<” por el sistema ya que está próximo un número.

El contraejemplo que se encuentra en la oración “te echaremos de menos”.

Aquí no se podría aplicar la regla ya que no hay números cerca y por lo tanto no se trata de ningún operador matemático.

Otra funcionalidad que tiene este apartado es la de transformar términos que indiquen temporalidad por sus fechas correspondiente, esto se traduce a que palabras como “hoy”, “mañana” y “pasado mañana” son transformadas en su correspondiente fecha en un formato estándar.

El principal problema que presenta es la desambiguación en el término “mañana”. Por un lado mañana como sustantivo indica el tiempo entre el amanecer y el mediodía y por otro lado como adverbio significa el día siguiente al de hoy. Para desambiguarlo se recurre a nuestra lista de datos estructurados etiquetados. En el caso de adverbio se encontrará como dato válido mientras que en el caso de sustantivo se encontrará como nombre común porque por lo general le precederá de un artículo determinado.

Ejemplo:

- Mañana por la mañana nos vamos de compras.

  Adverbio temporal. Nombre común femenino.

  La primera aparición del término mañana se transformaría en una fecha del tipo 15/11/11.

  La segunda se cambiaría a un rango de horas desde las 06:00-12:59.

La última función que realiza es detectar que si aparece el término “tarde” comprueba que la hora está en el formato adecuado, por ejemplo: son las 5 de la tarde → son las 17:00.

3.4 Modelos

El sistema permite cargar información específica de un cierto dominio de aplicación para hacer el análisis más específico. Como se detallará más adelante en las pruebas se han desarrollado dos modelos: viajes y universidad.

El sistema presenta la posibilidad de no tener que pasar por ningún dominio, pero el resultado es previsiblemente peor ya que con los modelos de dominios se acota y optimiza la salida. Este subsistema contará con una parte de filtros y otra parte perteneciente a los propios modelos.

El cometido de los filtros es "limpiar" la entrada y la salida de los dominios, así conseguiremos reducir el tiempo de procesamiento, mejorar las prestaciones y se evita la introducción de ruido en el sistema.

3.4.1 Filtros

Se ha diseñado un sistema con dos filtros, el primero se denomina filtro estándar y el segundo filtro de dominio.

El primer filtro elimina las palabras que no son necesarias para el dominio, estas palabras fueron etiquetadas anteriormente con el etiquetador. El filtrado se hace por el tipo de palabra que es cada una de ellas, aprovechando que están etiquetadas. En caso que no se haya seleccionado ningún dominio, éste será el único filtro por el que pasen los datos.

El segundo filtro trata de un filtro de dominio, esto quiere decir que una vez el sistema haya pasado por algún dominio se filtran palabras que pueden ser sobrantes. Este filtrado se hace comprobando los valores con una lista preestablecida, si coincide algún valor éste es eliminado de la estructura de datos.

A efectos de validación del sistema, se han desarrollado los dominios Universidad y Viajes, aunque el sistema permite cargar cualquier fichero con otros modelos adicionales.

3.4.2 Dominio Universidad

Este dominio es el que particulariza la salida en caso que se trate de consultas relacionadas con la universidad. Un ejemplo del tipo de consultas que puede recibir el dominio es: *¿Quién imparte la asignatura de Cálculo?*

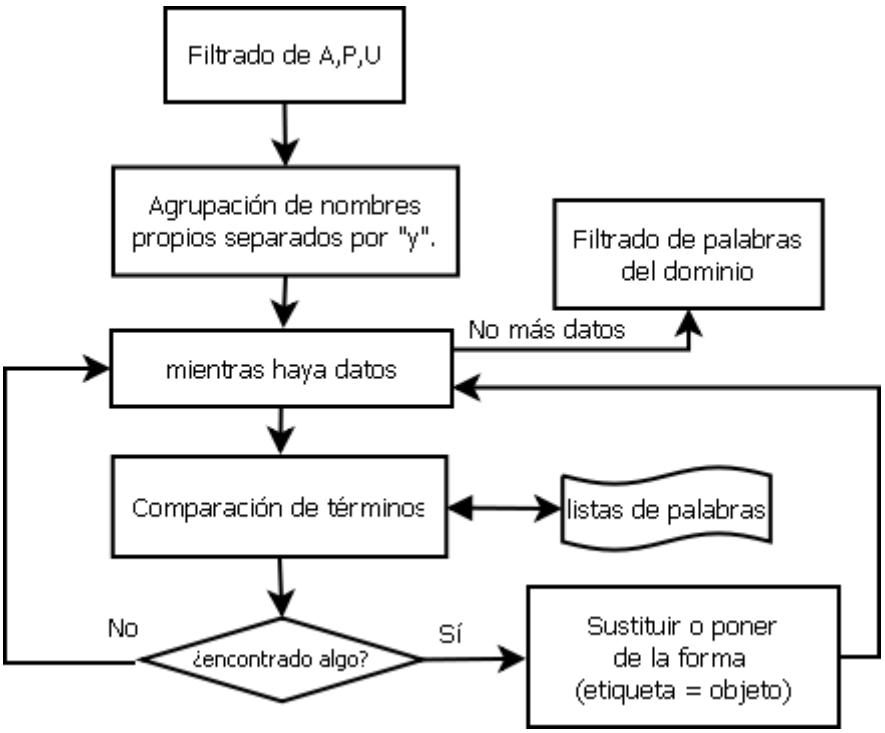


Imagen 10. Esquema dominio Universidad

El sistema como primera medida filtra las palabras que no le son útiles en este dominio. Estas palabras son artículos, preposiciones y palabras innecesarias, que el sistema las representa como A, P, U respectivamente.

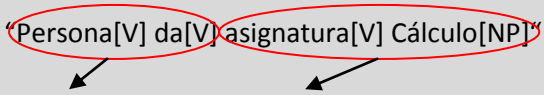
Este dominio cuenta con un agrupador de nombres propios en caso que estén separados por "y", esto se debe a que hay un alto número de asignaturas que se concatenan formando un único nombre propio y no dos como inicialmente trataría el sistema.

Hay que decir que el dominio cuenta con varias listas de palabras; términos para etiquetar más específicamente los objetos y términos que una vez procesada la información serán inútiles, es decir que podrán suprimirse tras el procesado. Estas listas pueden verse en el Anexo.

Las etiquetas para que sea lo más estándar posible se traducen a inglés. Simplemente se reemplaza la etiqueta en español y se pone en inglés.

La salida de este bloque devuelve una nueva estructura de datos con objetos identificados de la forma: "(etiqueta = objeto)". Antes de obtener la salida el sistema pasa por un filtro de dominio para eliminar los términos innecesarios.

Ejemplo:



Profesor[V] (asignatura = Cálculo)[NP] => (subject = Cálculo) (transformación en el dominio)

Profesor[V] (subject = Cálculo)[NP] (datos a la salida del dominio)

3.4.3 Dominio Viajes

En este dominio se particulariza la salida para un entorno relacionado con viajes. Un ejemplo de consulta que se adapta a este dominio puede ser: *Quiero información de vuelos para mañana.*

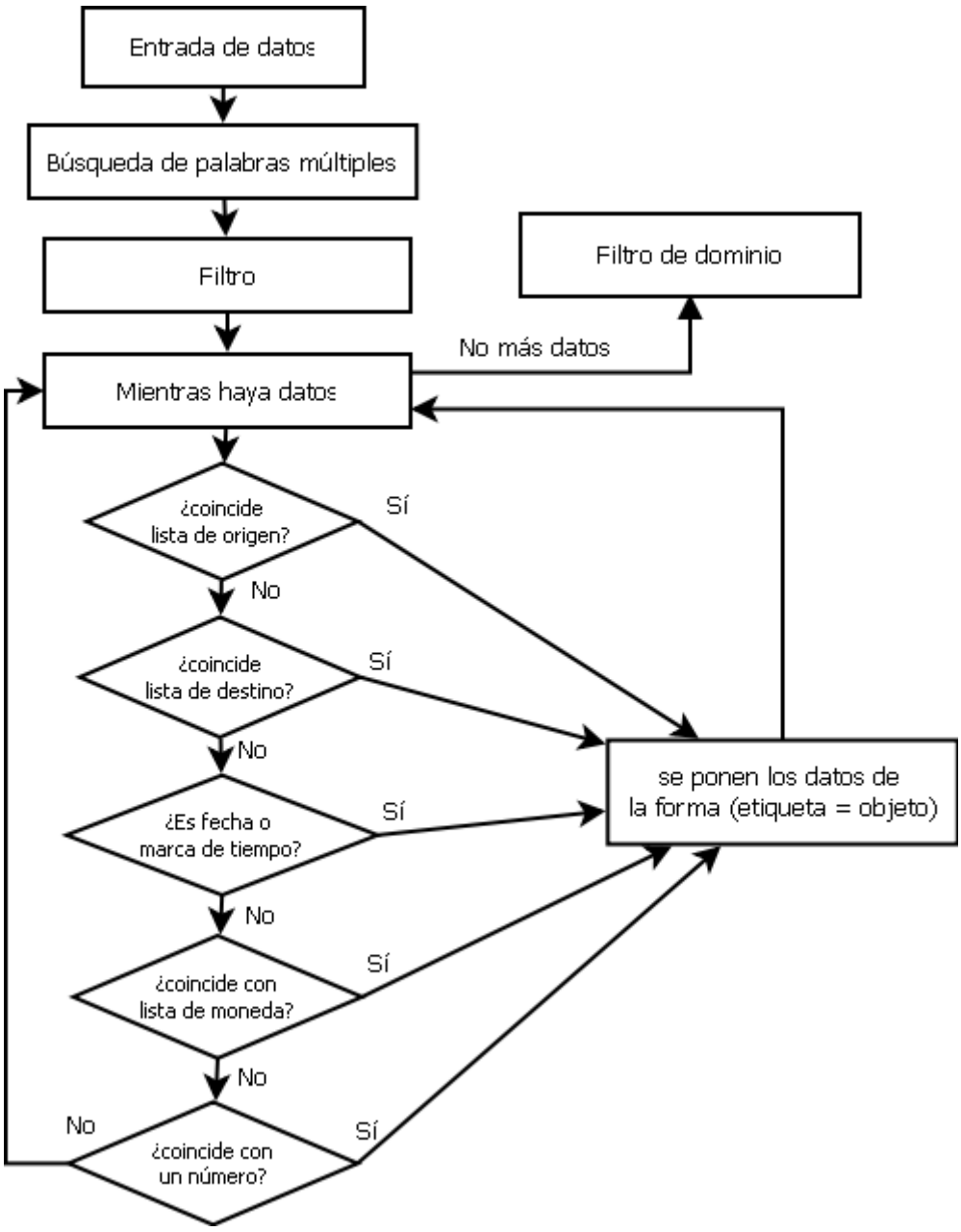


Imagen 11. Esquema dominio Viajes

El funcionamiento del dominio es parecido al dominio Universidad. En primer lugar se buscan palabras que pueden ser consideradas como únicos términos, multipalabras. Esto se hace para evitar problemas de interpretación del resultado. El sistema continúa con un filtrado de artículos y términos innecesarios, lo cual el sistema representa A|U respectivamente. En este caso no se eliminan las preposiciones ya que, en este dominio, pueden indicar procedencia, destino, etc.

El objetivo de este dominio es identificar términos para poder discriminar lo que es etiqueta de lo que es objeto y representarlo de manera adecuada. La búsqueda para identificar estos términos se hace mediante la comparación con unas listas, disponibles en el Anexo. Estas listas, con términos predefinidos, establecen las nuevas etiquetas. Las nuevas etiquetas caracterizan el objeto para ponerlo de la forma "(etiqueta = objeto)". Las etiquetas que pueden tener presencia son: destino, origen, fecha de salida, fecha de regreso, moneda, precio, hora de salida y hora de llegada. Todas estas etiquetas se mostrarán en inglés.

El dominio procesa la información tras el filtro y va comprobando posición a posición si coincide con los datos de alguna lista. En caso afirmativo se cambia la etiqueta a la forma "(etiqueta = objeto)".

Por último antes de volcar la información para el siguiente proceso, se pasa por un filtro para eliminar las palabras que a partir de ese momento sean innecesarias.

Ejemplo

Voy[V] desde[P] Madrid[NP] a[NP] Valencia[NP] (entrada en el dominio)

(origin = Madrid)[NP] (destination = Valencia)[NP] (salida del dominio)

3.4.4 Salida

La salida puede ser de dos maneras; texto plano contenido o una estructura XML, para ello el usuario dispondrá de opciones para seleccionar la salida que más le convenga.

El subproceso que gestiona la salida recibirá una estructura de datos, para poder imprimir los datos y representarlos de una forma más "amigable" al usuario, el sistema convierte estos datos en una cadena de palabras.

Para la salida de texto plano el diseño que se ha realizado consiste principalmente en separar los términos con "AND" y posteriormente imprimir en pantalla toda la cadena de palabras relevantes que ha extraído el sistema.

Para la salida en XML como es un lenguaje formal se requiere de cierta lógica para transformar lo que le llega de la salida de los filtros.

El proceso consiste en transformar la salida de los filtros a una estructura XML, para ello se colocan todas las etiquetas correctamente y se transforma la forma (etiqueta = objeto) a la forma adecuada de XML que es “<etiqueta>objeto</etiqueta>”. Para el caso que no se seleccione ningún dominio no se garantiza que se devuelva en XML ya que es en los dominios donde más se particularizan los datos.

4 Implementación del sistema

En este apartado se va a explicar con mayor detalle las partes del código en el que se ha desarrollado la aplicación, además se explicaran las decisiones que se han tomado referentes a detalles de implementación como por ejemplo los lenguajes de programación utilizados.

4.1 Desarrollo de la aplicación

Esta aplicación ha sido desarrollada en PHP por los motivos que se explicarán en el siguiente apartado. Para ejecutar el programa se debe disponer de un servidor PHP preinstalado o se puede usar el programa WAMP Server o XAMP Server, en el caso de Windows o Linux respectivamente, ya que en ambos casos tanto la instalación como su uso son muy sencillos.

La aplicación consta de seis ficheros:

1. **query.html:** este es el fichero encargado de solicitar los datos al usuario mediante un formulario, seguidamente llamará a la función QueryParserDemo.php
2. **QueryParserDemo.php:** este fichero recoge los datos del formulario, muestra la página HTML con la salida, representa el XML y llama a la función `queryParser.inc`
3. **queryParser.inc:** este fichero contiene los métodos de procesamiento del texto para extraer y etiquetar la información adecuadamente. También se incluyen las funciones para mostrar la información final de los datos.
4. **ES_tags.inc:** este fichero es el que contiene la mayoría de listas de palabras que se utilizarán para el etiquetado. En el Anexo (listas generales) se pueden observar las listas de palabras de este archivo.
5. **universityDomain.inc:** este fichero contiene las funciones necesarias para particularizar el dominio Universidad.
6. **travelDomain.inc:** este fichero contiene las funciones necesarias para particularizar el dominio Viajes.

A continuación se muestra un esquema de cómo será el sistema.

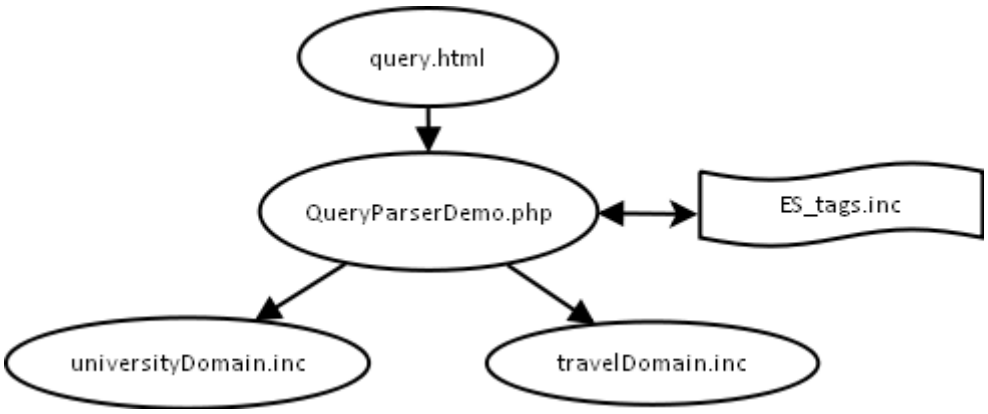


Imagen 12. Esquema de clases del sistema.

4.1.1 PHP

A continuación se mostrarán las razones por las que se ha programado en PHP:

- Es un lenguaje multiplataforma.
- Está orientado a aplicaciones web dinámicas.
- El código fuente es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en este lenguaje sea segura y confiable.
- Posee una amplia documentación en su página oficial [15], entre la cual se puede destacar la explicación y ejemplificación de todas las funciones de este lenguaje.
- Es libre, por lo que es una alternativa de fácil acceso para todos los desarrolladores.
- Permite aplicar técnicas de programación orientada a objetos.
- El programador puede aplicar en su sistema cualquier técnica de programación y desarrollo que le permita escribir código ordenado, estructurado y manejable.
- Tiene la definición de arrays asociativos: gracias a ellos el manejo de objetos es mucho más sencillo, estos arrays son los que se utilizan cuando se hace referencia a una estructura de datos. Además de facilitar la tarea de procesado de diferentes ficheros, lo cual facilita gran parte del trabajo de este proyecto.
- Expresiones regulares: gracias al potencial que tienen las expresiones regulares en PHP es mucho más sencillo extraer información de cadenas de texto.

4.1.2 WAMP

Para implantar el sistema se ha utilizado la aplicación WAMP Server [16]. A continuación se explicará su funcionalidad y la manera de utilizarlo para este caso.

Wamp Server es una plataforma para el desarrollo Web en Windows. Se le permite desarrollar aplicaciones Web dinámicas utilizando el servidor Apache 2 [17], el lenguaje de programación PHP y una base de datos MySQL [18]. También cuenta con PHPMyAdmin para facilitar la gestión de bases de datos.

Como se puede ver en su página web, Wamp Server se instala fácilmente y permite su uso de una forma muy intuitiva para configurar rápidamente.

A diferencia de otras soluciones, Wamp Server permite reproducir fielmente el servidor de producción. Una vez esté instalado se puede añadir tantas versiones de Apache, MySQL y PHP que se desee.

Para este proyecto en concreto la utilidad que proporciona sobre bases de datos no ha sido necesario utilizarla.

Tiene varias funcionalidades como la de gestionar servicios de Apache y MySQL, pasar de modo línea a fuera de línea (abierta a todos o limitada a localhost), instalar y cambiar la versión de Apache, MySQL y PHP, manejar las opciones de configuración para los servidores, acceder a los logs, acceder a los archivos de configuración y configuración de alias.

Ahora se pasará a explicar paso por paso cómo integrar el código del sistema con este software para poder llegar al aspecto que tiene finalmente mostrado en el Capítulo 5.

Al instalar Wamp Server se crea un directorio “www” automáticamente (normalmente en C:\wamp\www). Para empezar se debe crear un subdirectorio para el proyecto y guardar en él los archivos PHP que se han generado, en este caso el subdirectorio se llama “lengNatural” y en él se encuentran los ficheros PHP necesarios para la creación de la aplicación.

Imagen 13. Directorio www de WampServer

Para visualizar el proyecto en el navegador se puede proceder de dos maneras. La primera es haciendo clic en el enlace de "Localhost" que aparece al pulsar el botón de Wamp Server (ver imagen 12) a partir del cual se llegará a la siguiente página (imagen 13).

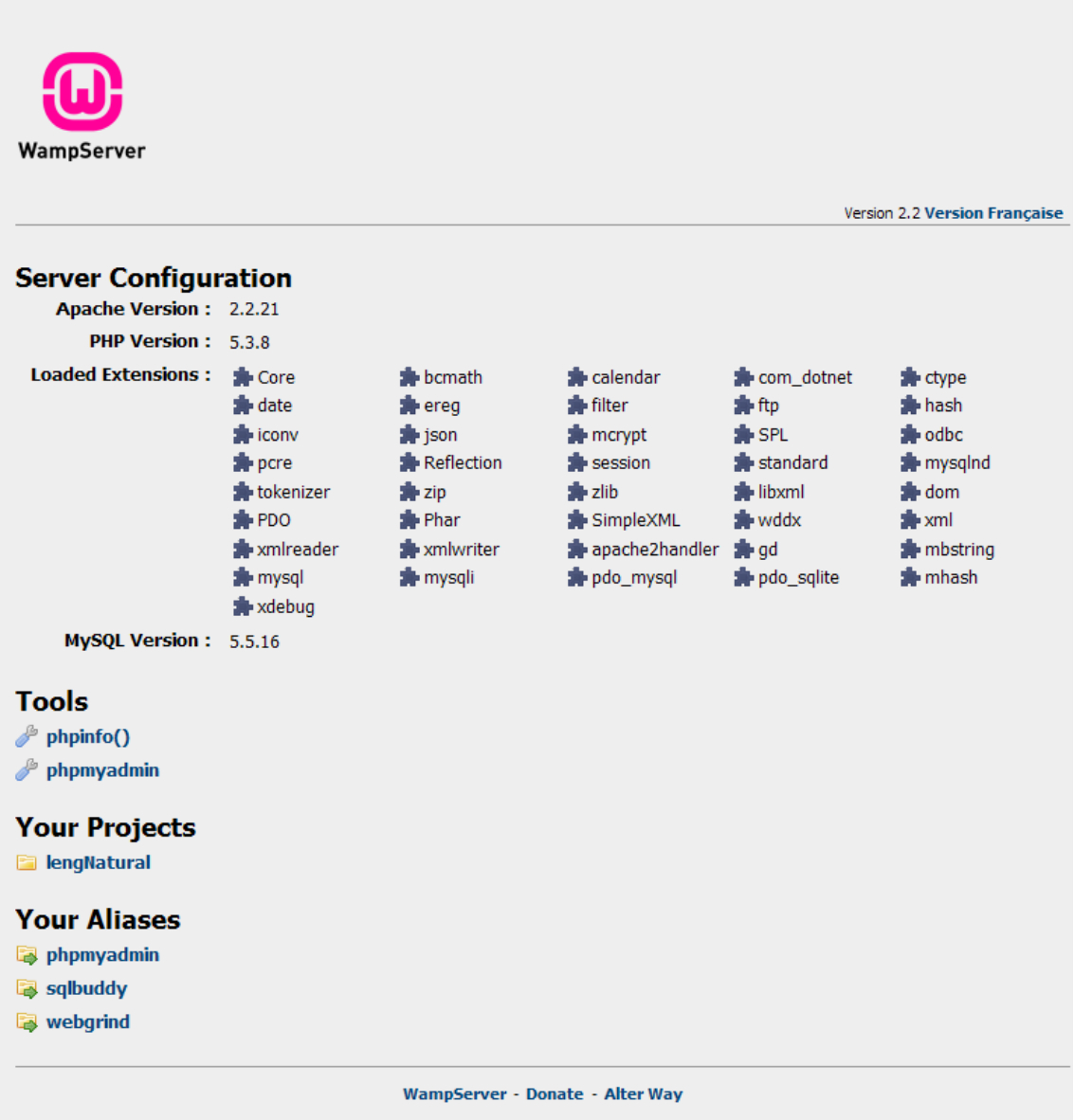


Imagen 14. Página de localhost de WampServer

La segunda forma es introduciendo la URL en el navegador directamente. En este caso pondremos en el navegador http://localhost/lengNatural/query.html y con ello accederemos a la aplicación que se ha creado.

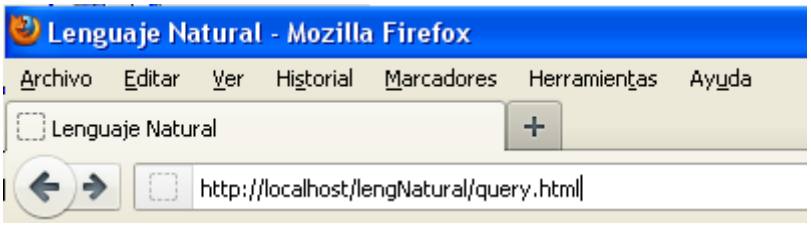


Imagen 15. Introducción de URL en el navegador

Una vez realizados estos sencillos pasos se podrá utilizar la aplicación en cualquier ordenador.

4.2 Implementación de la aplicación

En este apartado se va a explicar cómo se ha desarrollado la aplicación por niveles de bloques de funcionalidad. Se irá haciendo una relación de los métodos que se entran dentro de cada bloque, así como la clase que los contiene.

4.2.1 Bloque I: Analizador de la consulta

En este bloque intervienen las clases `QueryParserDemo.php` y `queryParser.inc` y el fichero `query.html`.

En el fichero `query.html` se encuentra un formulario con dos botones para seleccionar la salida (xml o texto), otros dos botones para seleccionar el dominio (Viajes o Universidad) y una entrada de texto que es donde se introducen los datos de la consulta. Es una página html muy sencilla, la cual se mostrará en el apartado 5.1 Presentación del sistema.

La clase `QueryParserDemo.php` es la encargada de recoger los datos del formulario y llamar a la función `QueryParser` de la clase `QueryParser.inc` cuando se hayan recogido los datos. Esta función es una especie de método principal, encargado de llamar al resto de funciones. La llamada se hará pasándole dos argumentos, los datos recogidos y el dominio.

```
function readForm($name, $default = '') {
        if(isset($_POST[$name]))
                return stripslashes($_POST[$name]);
        elseif(isset($_GET[$name]))
                return stripslashes($_GET[$name]);
        else
                return $default;
       }
```

Los datos están enviados mediante el método POST, no obstante se deja el método GET para posibles nuevas implementaciones o mejoras.

En este bloque interviene un método de la clase `QueryParser.inc` llamado `queryToArray` en el que le llegan los datos de la consulta. El objetivo es el de procesar la información para dejarla en una estructura de datos o array. Antes de pasar los datos al array se deben "limpiar", es decir, eliminar puntuación, corregir los meses y juntar las multipalabras, esto nos evitará ruido en el sistema.

```
function queryToArray($datos) {
        global $ES_mw;
        global $months;
        $arrSignosPuntu = array('@','#','%','^','&','*','(',')','=','+','[',']',';',';','¿','?','.');
        if(preg_match('/[a-z]?,[a-z]?/i',$datos)>0) { //esto evita el problema de la coma detrás de una palabra.
                $datos = str_replace(',',' , ',$datos);
        }
        foreach ($arrSignosPuntu as $clave => $value) {
                $datos = str_replace($value,' ',$datos);
        }
        // Multiwords. En el espacio de una palabra se introducen las correspondientes...
        for($i=0; $i<count($ES_mw); $i++) {
                $aux = str_ireplace('_',' ',$ES_mw[$i]);
```

```
                $datos = str_ireplace($aux,$ES_mw[$i],$datos);
        }
        preg_match_all('/([^ ]+) */', $datos, $aSalida); //datos en el array sin espacios
        $datosArr = $aSalida[1];
        for($i=0; $i<count($datosArr); $i++) {
                $datosArr[$i] = str_ireplace('_',' ',$datosArr[$i]);
                if(isset($months[strtolower($datosArr[$i])]) ) {
                        if(isset($datosArr[$i+1]) && $datosArr[$i+1]
!=ucfirst($datosArr[$i+1])){ //si la siguiente posición tiene otro nombre en mayúsculas se
trata de un nombre propio sino es un mes.
                                $datosArr[$i] = strtolower($datosArr[$i]);
                        }
                }
        }
        return $datosArr;
}
```

Para dejar los datos limpios contamos con una lista llamada “$arrSignosPuntu” en la cual se incluyen los signos de puntuación que pueden estorbar en el sistema, para que esto no ocurra se buscan los signos y en caso de aparecer se eliminan.

Para evitar que una coma se adhiera a una palabra en el sistema se opta por poner un espacio entre la coma y el término.

En el caso de las multipalabras se ha optado por buscar en la lista de palabras si algunas de las palabras de la consulta coincide, si se encuentra dicha cadena de palabras se reemplaza de la forma en la que está en la lista, ya que viene con guiones bajos para indicar que se trata de una multipalabra y el sistema posteriormente no encontrará espacios y la adjudique a una única posición del array. Una vez en el array ya se puede volver a poner los espacios en su correspondiente sitio ya que no afectará a la estructura al estar en una única posición.

Para pasar de una cadena de texto a un array en PHP es muy sencillo, se llama a la función `preg_match_all` y mediante expresiones regulares buscamos los espacios para separar las

palabras. El resultado de esta operación es que cada término encontrado entre espacios nos lo devuelve en una posición del array.

Por último se encuentra el corrector de meses. Hay disponible una lista de los meses en minúsculas, entonces se comparan los datos de la consulta convertidos a minúsculas con la lista y en caso de coincidir se cambia al valor en minúscula. Hay una pequeña excepción y se hace para evitar que nombres propios se etiqueten como mes, para ello tiene que venir la primera letra del nombre del mes en mayúscula y la siguiente palabra también tiene que llevar su primera letra en mayúscula.

Por último, esta función devuelve un array de datos para ser procesado por los siguientes módulos.

4.2.2 Bloque II: Etiquetador

En este bloque solamente intervienen la clase `QueryParser.inc` y el archivo de listas de palabras `ES_tags.inc`. Los métodos de este bloque son: `tagName`, `tag`, `postTag`.

Función tagName

La función `tagName` recibe los datos en forma de array y trata de buscar los nombres propios, si encuentra alguno los guarda en una variable global, que es un array asociativo. La implementación de las búsquedas se ha hecho mediante bucles “for” tanto para los nombres propios como los títulos entre comillas. La salida de este método devuelve el array de datos modificado en caso que hubiese algún nombre propio ya que habría que recolocar las posiciones.

```
if(mb_ereg_match("^[ÁÉÍÓÚ]|^[A-Z]",$datosArr[$i])) { //se ha encontrado la primera palabra con mayúscula
        $flagMayuscula = true;
        $nPropio[$i]=array('data'=>$datosArr[$i],‘type'=>'NP','source'=>$datosArr[$i]);
}
```

Este es un pequeño fragmento de código en el que se muestra como encontrar alguna palabra cuya primera letra sea en mayúscula. También se muestra como quedaría la estructura del array tras ser etiquetado como nombre propio.

Ejemplo

Array de datos	Array nombres
Voy	
de	
Madrid	Madrid
a	
Valencia	Valencia

Función tag

La función `tag` recibe como parámetros los datos de la salida de `tagName`, también se tienen disponibles los meses y los nombres propios etiquetados por el módulo anterior en un array.

Para realizar el etiquetado correctamente y eficientemente la búsqueda de las palabras de las listas se hace mediante índice porque supone un mayor rendimiento y una reducción del tiempo de búsqueda. Para que esto sea posible las listas de palabras se pasan a arrays, esto es una ventaja a la hora de buscar por índice ya que PHP tiene una función que cambia el índice por el valor y el valor por el índice, dicha función se llama `array_flip()`.

Por lo tanto se pasa de tener una lista de palabras a un array de palabras cuyos índices son las propias palabras.

```
for($i=0; $i<count($datosArr); $i++) {
 //nombres propios, se busca por la misma posición
        if(isset($nPropio[$i])) {
                $etiquetas[] = $nPropio[$i];
        }
        //Se comprueba si está la palabra en la lista buscando por el índice, más rápido.
        //pronombres interrogativos
        else if( isset($arrQuestion[$datosArr[$i]])) {
                $etiquetas[]=array('data'=>$arrQuestion[$datosArr[$i]],'type'=>'Q',
                'source'=>$datosArr[$i]);
        }
        //palabras importantes
        else if( isset($arrImp[$datosArr[$i]])) {
                $etiquetas[]=
                array('data'=>$arrImp[$datosArr[$i]],'type'=>'I','source'=>$datosArr[$i]);
        }
}
```

En cada iteración se irá comprobando si existe el índice actual en el array de nombres propios, si esa posición existe quiere decirse que se ha encontrado un nombre propio, con lo cual se añade al nuevo array de etiquetas y se pasa a la siguiente posición. En caso negativo se comprueba si el dato de esa posición coincide con alguna de las palabras de las listas preestablecidas, de ser así y sabiendo de qué lista provenía, el término se puede etiquetar con los tipos preestablecidos. En caso de que no encontremos nada se le pondrá en el tipo de palabra la etiqueta válido(V).

Función PostTag

El método `postTag` es el encargado de realizar lo que se definió en el diseño como post etiquetado. Entre las funciones principales que realiza son la de cambiar las palabras que indiquen temporalidad, como pueden ser hoy, mañana y pasado mañana, a su fecha correspondiente. También cambia un término que denote lo mismo que un operador matemático, como por ejemplo mayor que, siempre y cuando haya un número próximo.

```
for($clave=0; $clave<count($etiquetas); $clave++) {
        if($etiquetas[$clave]['type'] == 'N' || $etiquetas[$clave]['type'] == 'D'
||$etiquetas[$clave]['type'] == 'T') {
                foreach($etiquetas as $clave2 => $value2) { //se busca la existencia de alguna
                cifra.
                        if(isset($ES_arrOp[$etiquetas[$clave2]['data']])) {
                                $etiquetas[$clave2]['data']=$ES_arrOp[$etiquetas[$clave2]['da
                                ta']] ;
                                $etiquetas[$clave2]['type'] = 'O';
                                break;
                        }
                }
        }
        switch($etiquetas[$clave]['data']){
                case 'hoy':
                        $etiquetas[$clave]['data'] = date('d-m-Y' ,mktime(0,0,0,date("m"),
                        date("d"), date("Y")));
                        $etiquetas[$clave]['type'] = 'D';
                        break;
                case 'mañana': //hay que distinguir si es mañana (temporal) de día o por la
        mañana (NC).
                        if(isset($etiquetas[$clave-2]['data']) &&isset($etiquetas[$clave-
1]['data']) && $etiquetas[$clave-2]['data'] == 'de' &&$etiquetas[$clave-1]['data'] == 'la') {
                        //por defecto :00
```

```
                                if(strpos($etiquetas[$clave-3]['data'],':') === false ){
                                        $min = ':00';
                                        $etiquetas[$clave-3]['data'] .= $min;
                                }
                                $etiquetas[$clave-3]['type'] = 'T';
                                unset($etiquetas[$clave-2]);
                                unset($etiquetas[$clave-1]);
                                unset($etiquetas[$clave]);
                                }
                                else if($etiquetas[$clave]['type'] == 'NC'){
                                        $etiquetas[$clave]['data'] = 'range_time = 06:00 - 12:59
';
                                        $etiquetas[$clave]['type'] = 'T';
                                }else{
                                        if(isset($etiquetas[$clave-1]['data']) && 'pasado' ==
$etiquetas[$clave-1]['data']) {
                                                $diasPosteriores = 2;
                                                $etiquetas[$clave-1]['type']='U';
                                        }else{
                                                $diasPosteriores = 1;
                                        }
                                        $strfecha = date('d-m-Y' ,mktime(0,0,0, date("m"),
date("d")+$diasPosteriores, date("Y")));
                                        $etiquetas[$clave]['data'] = $strfecha;
                                        $etiquetas[$clave]['type'] = 'D';
                                }
                                break;
 }
```

El procedimiento que se sigue es el habitual, se recorre el array de datos etiquetados que llega de la función `tag` y se buscan los operadores, si se encuentra se cambia el valor textual por el símbolo.

Siguiendo el proceso se llega a un switch, en el cual hay varias opciones, “hoy”, “mañana” y “tarde” aunque en el ejemplo anterior solo se muestra “hoy” y “mañana”. El caso de “hoy” es muy sencillo ya que simplemente se localiza “hoy” y se cambia por la fecha actual. En cuanto al caso del término “mañana” hay que mirar si es adverbio temporal o un sustantivo, una vez identificado, simplemente hay que reemplazar por lo que corresponda.

La aparición de la función unset() es para que el array siempre esté correctamente indexado y no haya incongruencias.

En ciertas ocasiones los datos sufrirán cambios para adaptar la salida a una forma lo más uniforme posible, éstos cambios se realizarán sin modificar la forma en que los datos se guardan, por tanto la salida que devuelve este método es un array.

4.2.3 Bloque III: Modelos

En este apartado se muestra la implementación de los filtros y la implementación de los dominios.

A continuación se van a mostrar los dos filtros, el primero se corresponde el filtro antes de entrar en algún dominio y el segundo se trata del filtro antes que el dominio devuelva la salida.

```
/************************Filtro estándar************************************/
//Array de etiquetas y el tipo de palabras a filtrar separadas por |
function filter($arrEti,$strFilter) {
	foreach($arrEti as $key => $value) {
		if(!(preg_match("/^($strFilter)$/",$arrEti[$key]['type']) > 0)) {
			$etiquetasFilter[] = $arrEti[$key];
		}
	}
	return $etiquetasFilter;
}

/************************Filtro dominio*************************************/
/**** Filtro por dominio, se eliminan palabras que sobran en ese dominio, preposiciones, palabras
```

```
innecesarias y artículos **********/
function filterDomain($etiquetas, $list) {//segundo filtrado, eliminar palabras relacionadas con el dominio.
        $auxi = array();
         foreach($etiquetas as $clave=>$value) {
                $etiquetas[$clave]['data']=preg_replace("/\b($list)\b/i","",$etiquetas[$clave]['data']);
                if( !(preg_match("/^(U|A|P)$/",$etiquetas[$clave]['type'])>0)
&&!empty($etiquetas[$clave]['data'])) {
                        $auxi[] = $etiquetas[$clave];
                }
        }
        return $auxi;
}
```

El primer filtro recibe por parámetro los datos etiquetados y una cadena del tipo "U|A|P" que indicará los tipos de palabras que se desean filtrar. Para filtrarlas se aplica la función `preg_match()` aplicando una expresión regular, los datos que superen el filtro, es decir los datos que no estén etiquetados de la forma en la que llegan en la cadena, son depositados en un nuevo array pero manteniendo todas las etiquetas.

El segundo filtro es más o menos parecido salvo que en vez de recibir una cadena con los tipos a filtrar, recibe un array que contiene una lista de palabras a filtrar, estas palabras son propias de cada dominio. Igual que el filtro anterior se llama a la función `preg_match()` con su correspondiente expresión regular, en caso que no se encuentre ninguna palabra de la lista el filtro es superado y puesto en un nuevo array.

Dominio Universidad

El dominio cuenta con una lista de palabras que indican diferentes aspectos: listas de palabras para determinar si es un profesor, una asignatura…, también cuenta con listas de términos que una vez procesados se pueden eliminar del sistema. Para ver la lista de palabras relacionadas con este dominio ver Anexo.

Este dominio se corresponde a la clase `universityDomain.inc`

```
        for($i=0 ;$i<count($arrEti);$i++) { //recorremos el array de etiquetas
                //sustituimos asignatura|profesor... por el ejemplo (asignatura = Cálculo)
                if(preg_match("/(".ES_SUSTITUTE.")
NP/",$arrEti[$i]['data'].''.(isset($arrEti[$i+1]['type'])? $arrEti[$i+1]['type']: ""))>0) {
                        $arrEti[$i]['data']          =          '('.translateTags($arrEti[$i]['data']).'
='.$arrEti[$i+1]['data'].')';
                        unset($arrEti[$i+1]);
                        $arrEti = array_values($arrEti);
                }
                //Sirve, por ejemplo para decir que una persona que imparte enseñanza es un
profesor-->(profesor = Paco)
                else if((preg_match("/(".ES_TEACH.")/i",$arrEti[$i]['data']))>0) {
                        foreach($ES_arrREP as $key=>$value) {
                                if(isset($arrEti[$i-1]) && $key == $arrEti[$i-1]['data']) {
                                        $arrEti[$i]['data'] = $value;
                                        $arrEti[$i-1]['type'] = 'U';
                                        break;
                                }
                        }
                }
                //fecha en el formato (date = fecha correspondiente)
                else if('D'== $arrEti[$i]['type']){
                        $arrEti[$i]['data'] = '(date = '.$arrEti[$i]['data'].')';
                }
        }
        return filterDomain($arrEti,ES_UNIVERSITY_FILTER);
}
```

En el código pueden observarse las diferentes listas de palabras, `ES_SUSTITUTE` y `ES_TEACH` para extraer algo de información útil y `ES_UNIVERSITY_FILTER` es la lista de palabras que se pasa al filtro junto con los datos.

Hay un bucle “for”, el cual es el encargado de recorrer el array de datos. Para extraer la información se compara un elemento con una lista de palabras, si la comparación es positiva se cambia el dato correspondiente a la forma “(etiqueta = objeto)”. Para que el array siga siendo coherente se borran las posiciones correspondientes.

Por último, se usa el filtro para separar la información útil de la que no lo es. Este método devolverá un array de datos etiquetados.

Dominio Viajes

Este dominio particulariza la salida para entornos de viajes. Cuenta con listas de palabras que le ayudan a decidir si es un destino, un origen, una fecha, etc. Para ver la lista de palabras correspondiente a este dominio ir a Anexo.

Este dominio se corresponde a la clase `travelDomain.inc`

```
if(preg_match("/^(".ES_DESTINATION.") NP/", $arrEti[$i]['data'].'
'.(isset($arrEti[$i+1]['type'])?$arrEti[$i+1]['type']: ""))>0) {
        $arrEti[$i] = $arrEti[$i+1];
        $dest = $arrEti[$i+1]['data'];
        $arrEti[$i]['data'] = '(destination = '.$dest.')';
        unset($arrEti[$i+1]);
        $arrEti = array_values($arrEti);
```

El fragmento de código mostrado se corresponde al proceso de captura de datos relacionados con destino, pero la forma de identificar el resto de posibles etiquetas es la misma. Para identificar un destino se busca mediante la función `preg_match()`, aplicando una expresión regular, se busca la combinación de un término de las listas de palabras más el tipo nombre propio. En caso afirmativo se pone de la forma “(etiqueta = objeto)” y se actualiza el array para que no se repitan los datos.

```
else if (preg_match('/\b(D|T)\b/',$arrEti[$i]['type'])) {
        $default = true;
        //fecha por un lado y hora por otro
        $aux = $arrEti[$i]['data'];
        if('D'== $arrEti[$i]['type'])
        $var = 'date';
 else
        $var = 'time';
        for($j=$i; $j>=0; $j--) {
                if(preg_match("/(".ES_DEPT.")/",$arrEti[$j]['data'])>0) {//esto será común para
        la fecha y la hora
                        $arrEti[$i]['data'] = "(departure_$var = $aux)";
                        $default = false;
                 }
                 else if(preg_match("/(".ES_ARRIVAL.")/",$arrEti[$j]['data'])>0) {
                        $arrEti[$i]['data'] = "(arrival_$var = $aux)";
                 $default = false;
                }
        }
}
```

El caso de fechas y horas es un poco distinto al resto porque solamente se compara si el tipo de palabra es fecha o tiempo y la forma de proceder es igual para ambos casos. Para determinar si se trata de un horario/fecha de salida o llegada se ha creado un array que vaya hacia atrás en búsqueda de palabras que puedan ayudar a decidir si se trata de salida o llegada.

Por último el método devuelve un array de datos etiquetados, previamente este array ha sido pasado por el filtro dominio para separar los datos válidos de los no válidos.

4.2.4 Bloque IV: Salida

La salida, como se planteó en el diseño, puede darse de dos maneras, en XML o en texto plano. Aquí intervienen las clases `queryParser.inc` que es la que contiene los métodos para la salida y la clase `QueryParserDemo.php` que es la que introduce las cabeceras para que se muestre correctamente.

Como primera medida en ambas salidas, se crea un nuevo array de una única posición a partir del array de datos etiquetados.

Salida en texto

Se trata de recorrer el array de datos e ir agrupando los términos. Para separar un término de otra se pone entre ellos "AND" para indicar que son términos distintos, en caso de una multipalabra aparecerá la palabra con espacios indicando que se trata del mismo término.

En caso que haya alguna negación de término se pone delante de ello "NOT" para indicar que es una negación. Este caso se puede extender a los operadores cambiándolos de signo. Ejemplo que se diga que "no es menor que" (NOT <) indica que es mayor (>).

La salida se devuelve como una cadena de términos para que en `QueryParserDemo.php` se agreguen las cabeceras correspondientes para que el usuario pueda verlo de una forma adecuada.

Salida en XML

Para hacer posible la salida en XML y que no haya ningún error todo debe estar correctamente etiquetado y correspondiendo a la estructura XML. En este apartado se introducen las etiquetas comunes y las de cada consulta el sistema las transforma de la forma "`(etiqueta = objeto)`" a `<etiqueta>objeto</etiqueta>`.

```
for($i=0; $i<count($aux); $i++) {
preg_match_all('/\(([^=]*)  ?=  ?([^\)]*)\)/',$aux[$i],$tag);//se detecta lo que hay en (destination = Madrid) ,por ejemplo.
        if (isset($tag[1][0])){
                for($j=0; $j<count($tag[0]); $j++) {
```

```
                        $str .= "\n";
                        $str .= '<'.$tag[1][$j].'>'.$tag[2][$j].'</'.$tag[1][$j].'>';
                }
        }else{//en caso de no identificar una etiqueta se transforma como un object.
                $str .= "\n".'<object><![CDATA['.$aux[$i].']]></object>';
        }
}
```

El proceso para cambiar la salida a una estructura de XML es mediante expresiones regulares. Estas expresiones regulares buscan el formato típico (“`etiqueta = objeto`”). Si es encontrado se cambia, en caso contrario se pone una etiqueta genérica denominada `objeto (object)` y entre las etiquetas se introduce el dato, quedando de la siguiente manera “`<object>valor<object>`”. La nueva estructura donde se almacenan todas las etiquetas junto a todos los datos es una cadena de texto. Esta cadena es enviada a `QueryParserDemo.php` para que se incluyan las cabeceras y que se muestre correctamente.

5 Pruebas del sistema

Este apartado de la memoria mostrará las diferentes pruebas que se han realizado del sistema implementado, con ello se pretende aclarar y ejemplificar el uso del programa que se ha diseñado.

5.1 Presentación del sistema

El sistema presenta una interfaz de entrada para procesar el texto, que facilita el manejo de la misma para el usuario final.

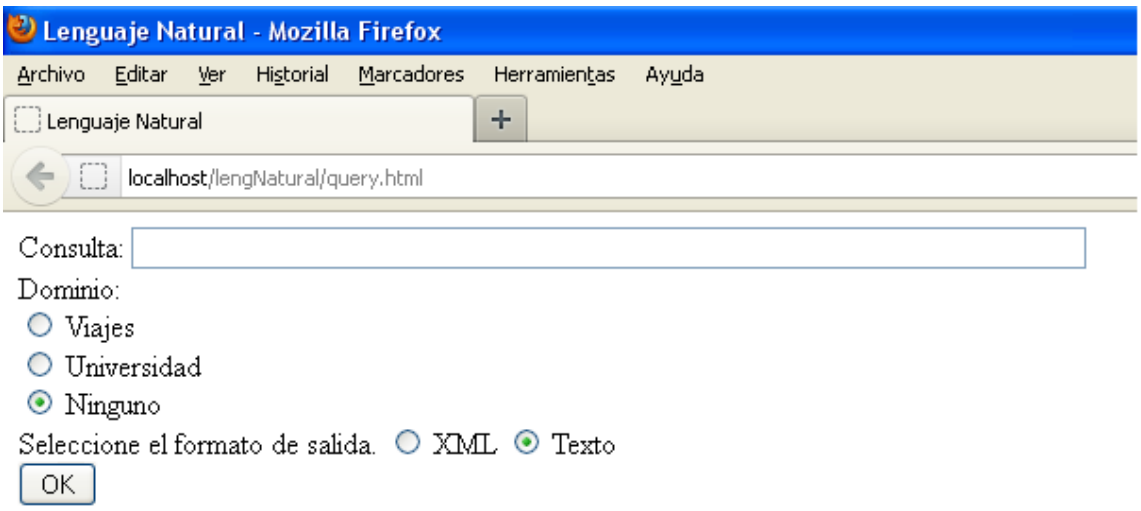


Imagen 16. Presentación del sistema

Como se puede observar, la interfaz cuenta con un aspecto muy sencillo que permite la elección de varias opciones. En primer lugar, cuenta con un campo denominado "Texto a procesar" donde se debe introducir el texto que se quiere procesar con este sistema. En segundo lugar, se muestran las opciones de selección de dominio, se puede seleccionar alguno de los dominios vistos anteriormente: Viajes, Universidad o ninguno de ellos. Por defecto la opción seleccionada es ningún dominio. Por último, tenemos la selección del formato de salida, se da la opción de elegir entre uno de los dos formatos, XML o Texto, este último es en forma de una cadena de texto, la opción que se encuentra por defecto es el formato de salida de texto.

Tras introducir el texto a procesar y seleccionar las opciones deseadas, se pulsa el botón de OK para que el sistema empiece a evaluar el texto.

En los siguientes puntos de este apartado de pruebas del sistema, se mostrarán diferentes ejemplos de consulta con el fin de intentar aclarar el funcionamiento del programa, y se evaluarán los resultados obtenidos en cada una de las consultas.

5.2 Ejemplo de consulta simple

A lo largo de este apartado las consultas se irán incrementando en dificultad, se mostrará para cada consulta los dos tipos de salida que hay.

El primer ejemplo que se va a ver es el procesado de lenguaje natural de una consulta simple, en la cual no hay seleccionado ningún dominio y se trata de una pregunta sencilla.

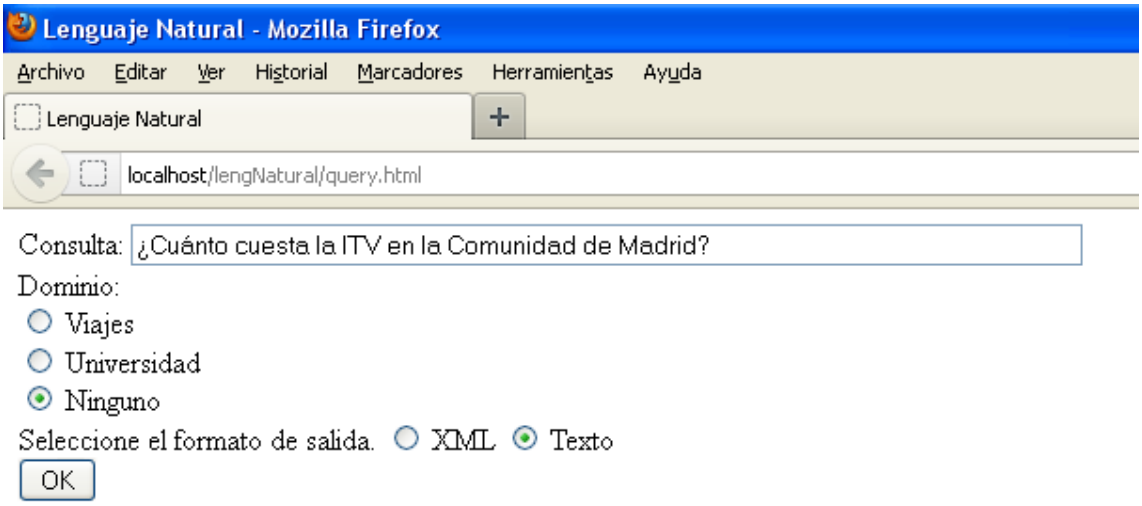


Imagen 17. Consulta número 1.

Como se observa en la consulta no se ha seleccionado ningún dominio y la salida es en texto, a continuación se muestra el resultado.

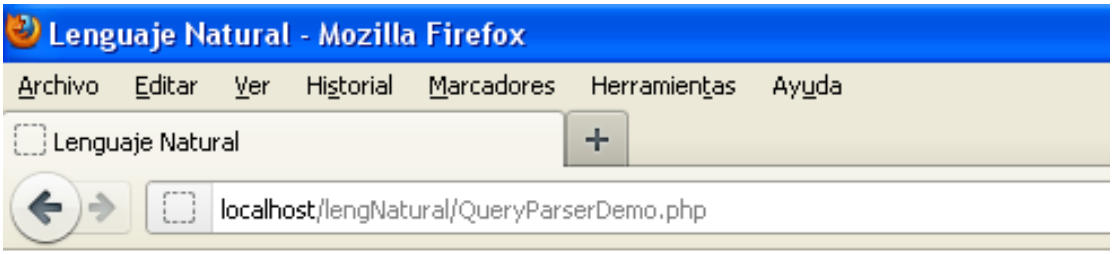


Imagen 18. Respuesta en forma de texto a la consulta número 1

Como puede observarse el proceso de lenguaje Natural ha eliminado, modificado y agrupado los términos de la entrada de la consulta. Así puede observarse que "cuánto cuesta" ha sido transformado a precio, "ITV" no lo borra del sistema y "Comunidad de Madrid" lo trata como un único término, el resto de palabras son eliminadas. Como separador se utilizan las letras "AND".

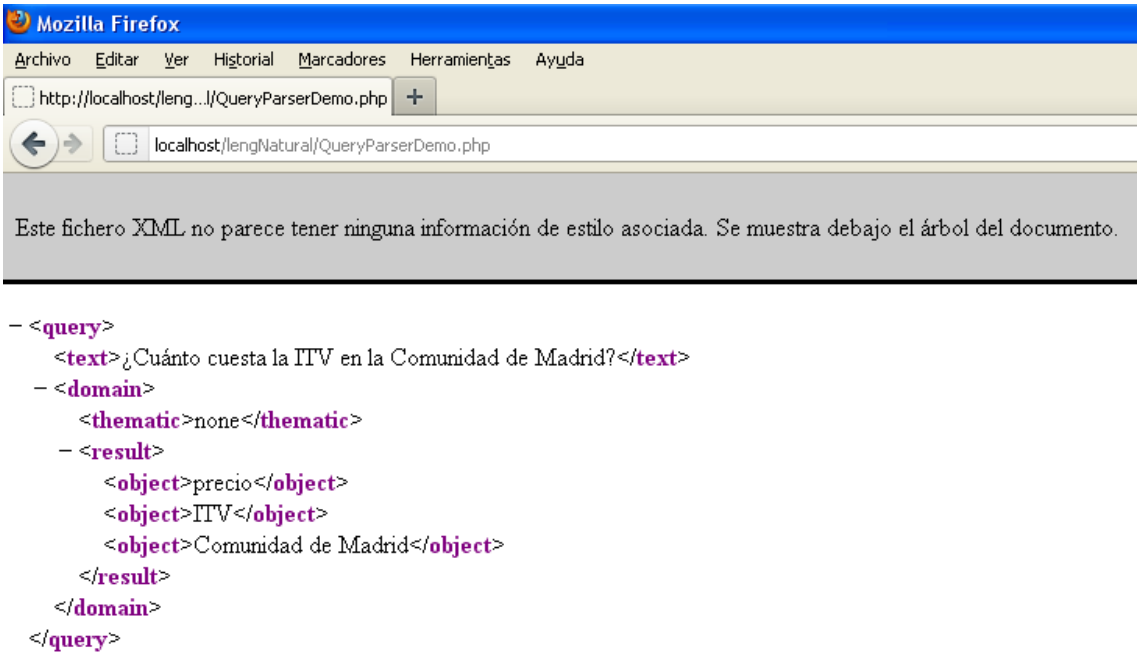


Imagen 19. Respuesta en formato XML a la consulta número 1

En la anterior imagen, se muestra la respuesta a los datos de entrada en forma XML, para ello previamente debe estar seleccionada la opción de salida como XML en la interfaz de entrada del sistema.

Los datos finales están representados en una estructura de XML. En primer lugar nos encontramos una etiqueta denominada “query” que engloba todo el conjunto de la consulta. Posteriormente está la etiqueta “text” en la cual se muestra el texto original introducido por el usuario. Siguiendo el orden descendente están las etiquetas “domain”, que engloba el resultado de la consulta, la etiqueta “thematic” que nos muestra si se ha seleccionado algún dominio específico, y la etiqueta “result” engloba los datos originales modificados y estructurados, por defecto se pone la etiqueta “object” salvo que el sistema reconozca y etiquete los térmninos. En caso que se hubiese identificado algo como fecha en vez de la etiqueta “object” aparecería “date”.

5.3 Ejemplo de consulta, dominio Universidad

En la siguiente imagen se expone una consulta del dominio Universidad, para que el sistema devuelva mejores resultados hay que seleccionar el dominio Universidad.

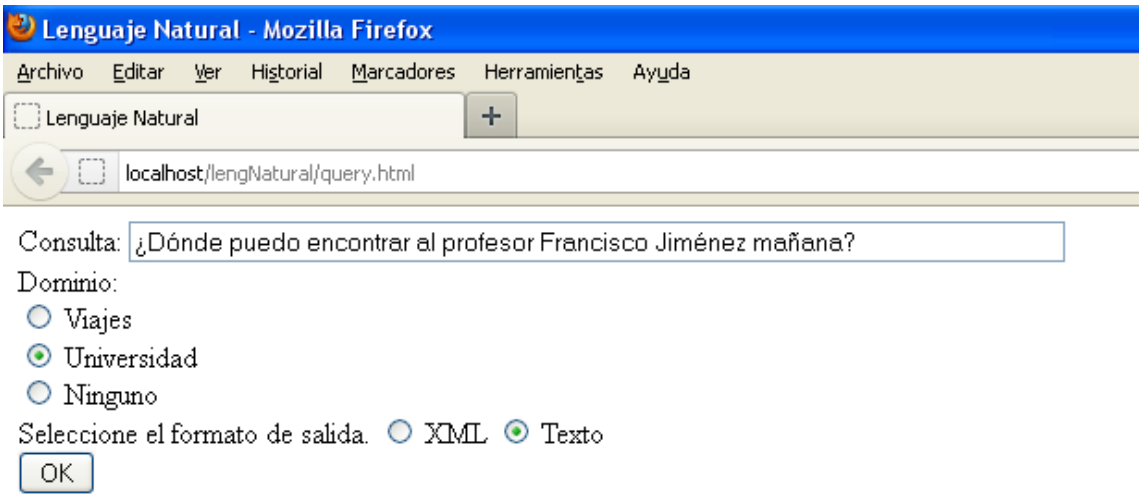
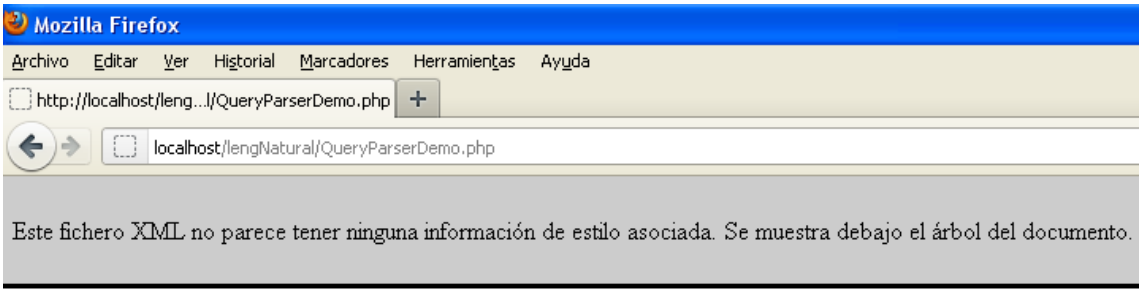


Imagen 20. Ejemplo consulta número 2

Este es un ejemplo de una consulta seleccionado el dominio Universidad. La salida como se puede ver es en Texto aunque primero se explicará la salida en XML.




```
− <query>
  − <text>
      ¿Dónde puedo encontrar al profesor Francisco Jiménez mañana?
    </text>
  − <domain>
      <thematic>university</thematic>
    − <result>
        <object>localización</object>
        <teacher>Francisco Jiménez</teacher>
        <date>06-01-2012</date>
      </result>
    </domain>
  </query>
```

Imagen 21. Respuesta en XML a la consulta número 2

En la imagen se muestra el resultado de la consulta, se puede observar que la temática es "university" y los resultados están categorizados salvo el término localización que es por el objeto el cual se pregunta. Se observa que el nombre "Francisco Jiménez" está en un único término y ha sido etiquetado como profesor, por lo que la palabra profesor del texto original ha sido eliminada automáticamente. Un caso parecido es el que pasa con el término "mañana", el sistema distingue entre si es un sustantivo o es un adverbio, en este caso determina que es un adverbio temporal y la palabra "mañana" es reemplazada por la fecha correspondiente y se etiqueta con "date".

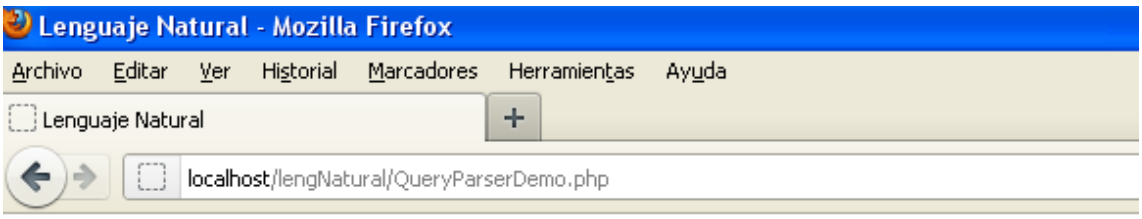


Imagen 22. Respuesta en formato de texto a la consulta número 2

Esta es la respuesta que se obtiene de la consulta número 2 en formato texto. Se puede ver que los datos obtenidos son los mismos que para la consulta con salida XML salvo que aquí no están etiquetados, simplemente tienen una similitud con las etiquetas de XML de la forma "(objeto = valor)" con lo cual se describe el objeto como término y detrás del igual se encuentra el valor.

Como en la anterior respuesta los términos vienen delimitados por las siglas "AND" y se muestra el texto original para compararlo con la respuesta.

5.4 Ejemplo de consulta, dominio Viajes.

En la siguiente consulta se expondrá como se aplica la temática de viajes.

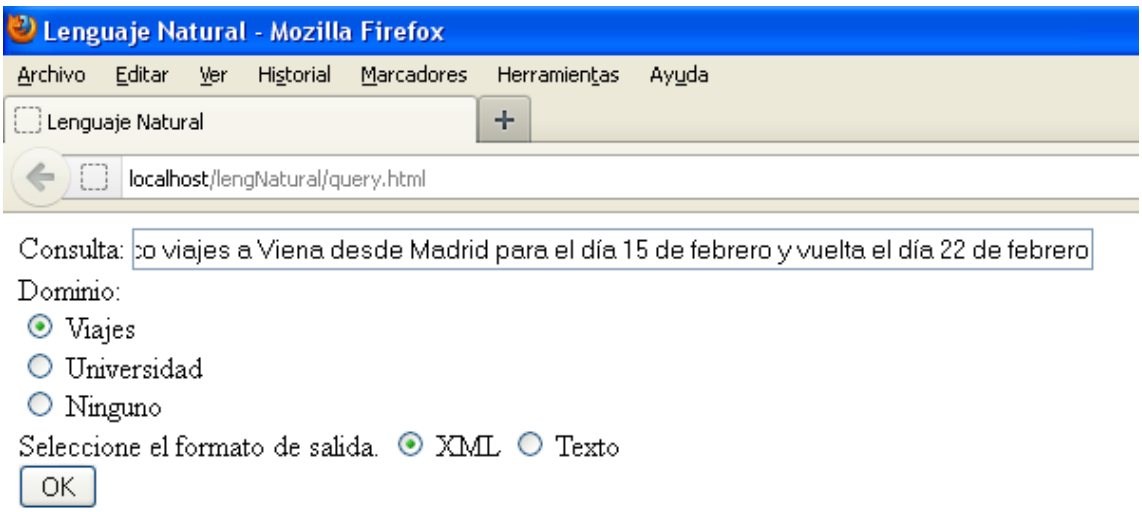
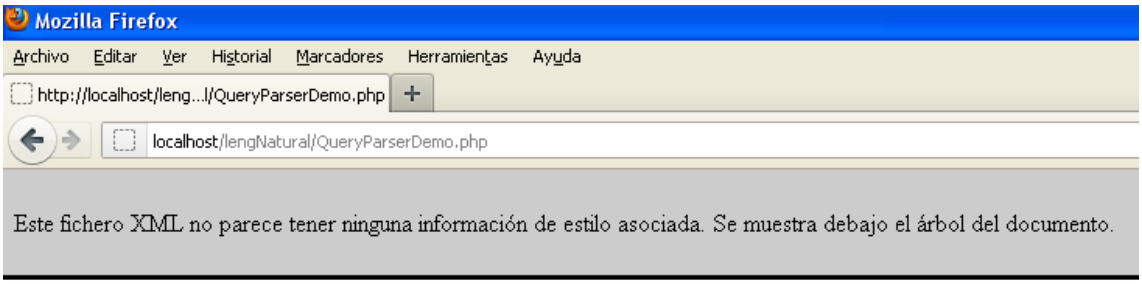


Imagen 23. Ejemplo consulta número 3

Para esta consulta se ha seleccionado el dominio Viajes y se ha seleccionado el formato de salida en XML. A continuación se observa la salida.




```
− <query>
  − <text>
      busco viajes a Viena desde Madrid para el día 15 de febrero y vuelta el día 22 de febrero
    </text>
  − <domain>
      <thematic>travel</thematic>
    − <result>
        <destination>Viena</destination>
        <origin>Madrid</origin>
        <departure_date>15-02-2012</departure_date>
      − <return>
          <destination>Madrid</destination>
          <origin>Viena</origin>
          <departure_date>22-02-2012</departure_date>
        </return>
      </result>
    </domain>
  </query>
```

Imagen 24. Respuesta en XML a la consulta número 3

En primer lugar como en el caso del dominio Universidad se observa el texto original introducido en el sistema. Luego la etiqueta “thematic” nos indica que se trata de una oración con relación a viajes. Siguiendo con el esquema nos encontramos con datos identificados.

El sistema aquí ha identificado objetos como destino, origen y fecha de salida, se puede comprobar que las ciudades introducidas han sido etiquetas de la manera correcta. En primer caso se pone Viena como destino y después de la etiqueta “return” viene identificado como origen. Ocurre el caso opuesto con la ciudad de Madrid, en primer lugar se encuentra como origen y posteriormente como destino.

También se puede observar que el sistema identifica las fechas en las que se quiere realizar el viaje. Y por último se puede observar que el sistema ha eliminado todas las palabras que no se consideraban necesarias como pueden ser las palabras vacías, preposiciones, determinantes…

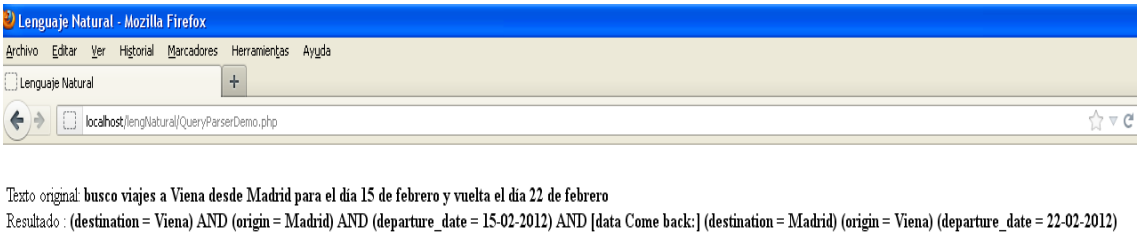


Imagen 25. Respuesta en formato texto a la consulta número 3

La salida se presenta igual que en el caso del dominio Universidad, se sigue la correspondencia de "(objeto = valor)" para los términos identificados. Aquí se puede observar que para diferenciar los datos de los que puede ser la ida en un viaje, se pone "[data Come back]" para identificar los datos que se corresponden al regreso.

5.5 Batería de pruebas

En este apartado se mostrará una serie de consultas que se han realizado en el sistema, se comentará si el resultado es bueno o malo, en caso que sea desfavorable se comentará el motivo.

Por motivos de simplicidad la salida del sistema se mostrará en texto.

Prueba 1
Texto original: **hoteles o casas rurales que no sean de alquiler íntegro en Castilla la Mancha o Castilla y León o Madrid** Resultado : **(hoteles OR casas rurales) AND NOT alquiler AND íntegro AND ((destination = Castilla la Mancha) OR (destination = Castilla)) AND (destination = León OR (destination = Madrid))**

Comentarios:

El sistema comete un pequeño fallo al no empaquetar Castilla y León en un único término. Esto es debido a que no el sistema no puede diferir si son dos nombres propios independientes o un único nombre propio. Una posible solución a estos casos excepcionales sería añadirlos a la lista de multipalabras. El dominio que se aplica es el de Viajes.

Prueba 2

Texto original: **quiero información de los horarios de los trenes de Valencia a Madrid de mañana por la mañana**

Resultado : **horarios AND trenes AND (origin = Valencia) AND (destination = Madrid) AND (departure_date = 07-06-2012) AND (departure_time = range_time = 06:00 - 12:59)**

Comentarios:

El dominio Viajes está activado y el resultado es correcto.

Prueba 3

Texto original: **¿Qué edad tiene Julia Roberts?**

Resultado : **edad AND Julia Roberts**

Comentarios:

No hay ningún tipo de dominio activado y el resultado es correcto.

Prueba 4

Texto original: **¿Quién impartirá la asignatura Cálculo el próximo curso?**

Resultado : **profesor AND (subject = Cálculo) AND próximo AND curso**

Comentarios:

El dominio que se ha aplicado ha sido el de Universidad y el resultado ha sido correcto. El sistema ha identificado que una persona que imparte en este dominio es un profesor.

Prueba 5

Texto original: **quiero saber la gente que nació en Madrid después de julio de 1950**

Resultado : **gente AND nació AND Madrid AND > 01-07-1950**

Comentarios:

No hay ningún dominio aplicado y la salida es correcta, identificando la fecha.

Prueba 6

Texto original: **Que ayudas se dan en la carrera Ingeniería Telemática?**

Resultado : **ayudas AND dan AND carrera AND Ingeniería Telemática**

Comentarios:

Está aplicado el dominio Universidad y la salida no es correcta, ya que incluye la palabra dan que en este caso no aporta información a la consulta.

Prueba 7

Texto original: **busco hoteles que no cuesten más de 200 euros y sean de 4 estrellas**

Resultado : **hoteles AND < (price = 200) AND (currency = euros) AND (price = 4) AND estrellas**

Comentarios:

El resultado se da con el dominio Viajes activado. La salida no es del todo correcta ya que identifica como precio la categoría de un hotel.

Prueba 8

Texto original: **noticias relacionadas con el mundo del espectáculo**

Resultado : **noticias AND relacionadas AND mundo AND espectáculo**

Comentarios:

La consulta no lleva ningún dominio asociada y la salida es errónea, ya que en este caso las palabras “relacionadas” y “mundo” no aportan valor semántico a la oración.

Prueba 9

Texto original: **¿Qué colegios hay en Valdemoro que sean Bilingües?**

Resultado : **colegios AND Valdemoro AND Bilingües**

Comentarios:

La respuesta es válida ya que se queda con los términos más relevantes de la consulta. La consulta no lleva dominio asociado

Prueba 10

Texto original: **necesito un coche de alquiler para mañana para ir a Lugo**
Resultado: **coche AND alquiler AND (departure_date = 07-05-2012) AND (destination = Lugo)**
.

Comentarios:

El dominio Viajes está activado y la respuesta es correcta.

6 Presupuesto

En este apartado se detalla el presupuesto del proyecto. En él se exponen las tareas, recursos y costes directos e indirectos que han sido necesarios para el desarrollo del proyecto sistema de extracción de información de una consulta en lenguaje natural.

6.1 Resumen de recursos y roles

Para el desarrollo del proyecto se tendrán en cuenta los siguientes perfiles:

- 1 Ingeniero Senior (Tutor). Ingeniero experto en PHP y Procesamiento de Lenguaje Natural. Será el responsable de asesorar y supervisar el proyecto y la documentación que conlleva. Actuará como consultor funcional.
- 1 Analista Programador (Alumno). El analista programador requerido debe tener un perfil técnico con conocimientos PHP. Las tareas que debe realizar son las siguientes: búsqueda de información sobre PLN, diseño de la aplicación, implementación de la aplicación, documentación del proyecto, pruebas del sistema.

6.2 Planificación del proyecto

Para la planificación del proyecto se han tenido en cuenta las siguientes fases:

Adquisición de conocimiento. Esta es la fase en la que se establecerán las bases del conocimiento en PLN e Ingeniería Lingüística así como las referencias tecnológicas a utilizar como son PHP, servidor WAMP, etc.

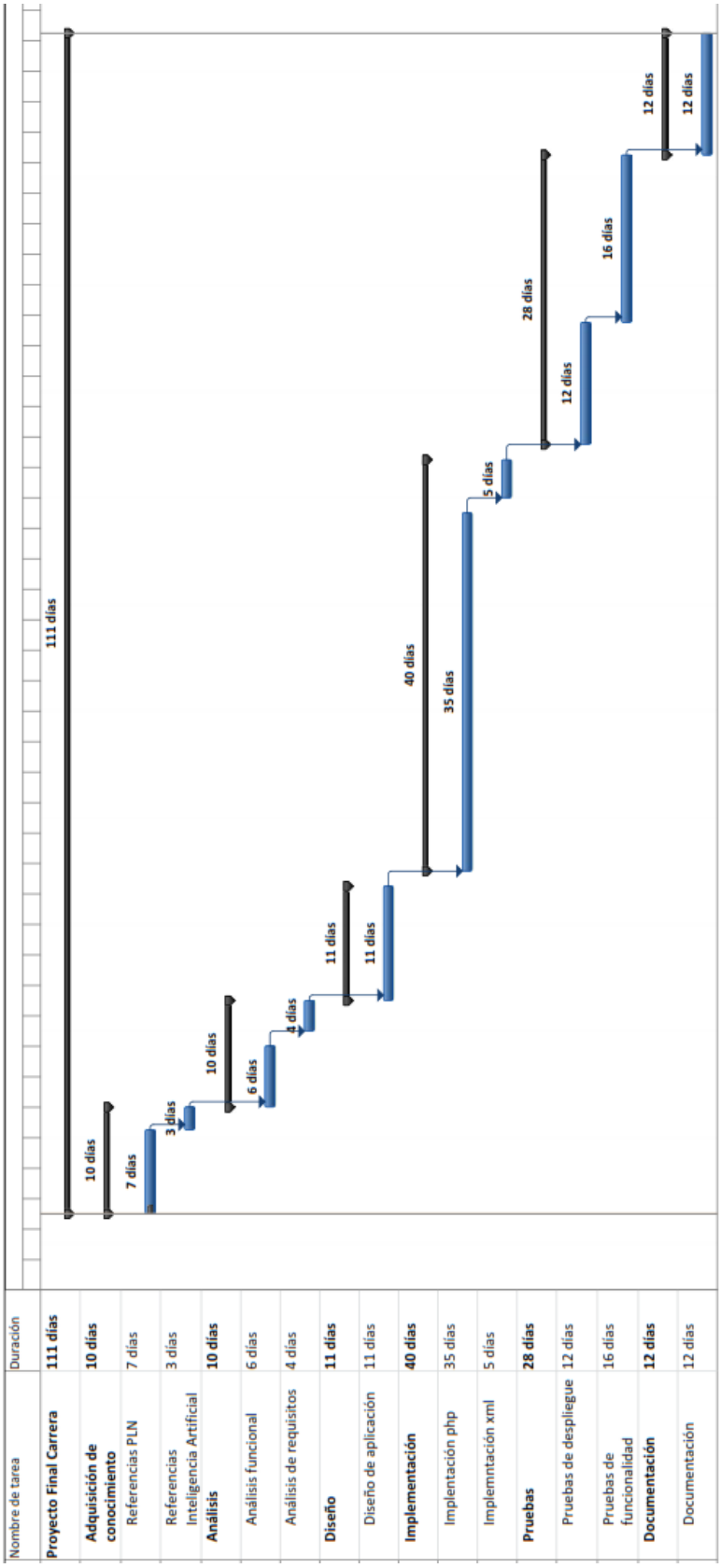
Análisis. En esta parte del proyecto se procederá a realizar el análisis funcional y de requisitos de la aplicación.

Implementación. En esta parte del proyecto se procederá al desarrollo del diseño expuesto en la fase de análisis. En primer lugar se desarrollará la parte para un funcionamiento genérico, posteriormente se desarrollarán los formatos de salida y por último los dominios específicos.

Pruebas. En el apartado de pruebas se probará el sistema, habrá dos tipos de pruebas: de despliegue y funcionales. Las pruebas de despliegue permitirán comprobar que el sistema funciona. Las pruebas funcionales se realizarán con una serie de consultas, valorando los casos

positivos de los que no se obtiene respuesta lo suficientemente válida como para sacar información.

Documentación. Esta es la última parte del proyecto y consiste en la realización de una memoria técnica.


Nombre de tarea
Duración
Proyecto Final Carrera 111 días
Adquisición de conocimiento 10 días
Referencias PLN 7 días
Referencias Inteligencia Artificial 3 días
Análisis 10 días
Análisis funcional 6 días
Análisis de requisitos 4 días
Diseño 11 días
Diseño de aplicación 11 días
Implementación 40 días
Implentación php 35 días
Implemntación xml 5 días
Pruebas 28 días
Pruebas de despliegue 12 días
Pruebas de funcionalidad 16 días
Documentación 12 días
Documentación 12 días
111 días
10 días
7 días
3 días
10 días
6 días
4 días
11 días
11 días
40 días
35 días
5 días
28 días
12 días
16 días
12 días
12 días

6.3 Costes directos

A continuación, la tabla muestra el resumen de la dedicación de recursos del proyecto desglosado por tareas y roles.

Tarea	Dedicación total (jornadas)	Dedicación total Ingeniero senior	Dedicación analista Programador
Referencias PLN	7	0%	100%
Referencias Inteligencia Artificial	3	0%	100%
Análisis Funcional	6	20%	80%
Análisis de Requisitos	4	20%	80%
Diseño de aplicación	11	30%	70%
Implementación php	35	0%	100%
Implementación xml	5	0%	100%
Pruebas de despliegue	12	10%	90%
Pruebas de funcionalidad	16	10%	90%
Documentación	12	25%	75%
Total (jornadas)	**111**	**11.1**	**99.9**

Para calcular el precio de la jornada dependiendo del perfil, se ha realizado mediante una ponderación basada en las tarifas de algunas consultoras en el desarrollo de PHP.

El coste de los recursos viene detallado en la siguiente tabla

Rol	Jornadas	Tarifa jornada	Total
Ingeniero senior (tutor)	11,1	360 €	3.996
Analista programador (alumno)	99,9	180 €	17.982
		Total	**21.978**

Al tratarse de software libre no se han requerido licencias de uso. Por lo tanto los costes que se han tenido han sido los procedentes de los ordenadores utilizados.

Descripción	Coste (€)	% Uso dedicado proyecto	Duración (meses)	Periodo de depreciación	Coste imputable
Ordenador portátil de desarrollo	800	95	11	60	95,33
Ordenador sobremesa ing. Senior	750	5	11	60	6.875
				Total	**102.20**

Teniendo en cuenta la duración del proyecto, la cual ha sido de once meses a media jornada, se han calculado los costes de amortización atendiendo a la siguiente fórmula de amortización.

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

Por lo tanto la suma de todos los costes directos asciende a 22.080,20.

6.4 Costes indirectos.

Se ha estimado que los costes indirectos son un 20% de los costes directos, resultando un total de 4.416,04.

6.5 Cuadro resumen del presupuesto

Total presupuesto del proyecto	
Total costes directos	22.080,20
Total costes indirectos	4.416,04
Total	**26.496,24**

7 Conclusiones y líneas de trabajo futuras

Para desarrollar este proyecto ha sido necesaria una documentación amplia sobre los sistemas de procesamiento de lenguaje natural. En el apartado 2, estado del arte, se han visto las principales partes que forman un sistema de procesamiento de lenguaje natural. Por otro lado, mediante los niveles del lenguaje se ha tratado de dar una imagen sobre la forma en la que se tiene que analizar el lenguaje humano, viendo los diferentes aspectos como pueden ser la sintaxis, semántica, morfología, fonología y pragmática.

Para este proyecto los niveles del lenguaje que han sido utilizados para el desarrollo de la aplicación han sido la sintaxis, la morfología y la semántica. Por otro lado, han quedado fuera de este proyecto los niveles del lenguaje que se corresponden con la pragmática y la fonología. La pragmática no se ha desarrollado en el proyecto por una cuestión de diseño, mientras que la fonología, queda fuera del ámbito de este proyecto al tratarse del estudio del sonido del lenguaje.

El mayor problema al abordar este proyecto ha sido la ambigüedad, tanto a nivel de término como de oración. A nivel de término viene dado por la presencia de homonimia, es decir, dos palabras se escriben igual, pero su significado es distinto y a nivel de oración, ésta puede ser interpretada de varias maneras. Un ejemplo muy típico es el siguiente: “Juan vio a María con el telescopio”. Como se aprecia el problema de la ambigüedad tiene un difícil tratamiento y es un problema que sufren los procesadores de lenguaje natural.

Teniendo en cuenta los anteriores aspectos se construyó el sistema para que tuviese un tiempo de acceso aceptable. El tiempo de procesamiento es muy bajo, inferior a 3 décimas de segundos en todas las pruebas que se realizaron. Estas pruebas consistieron en hacerle muchas peticiones al sistema monitorizando el tiempo de procesamiento.

Como se ha visto en el capítulo 5, pruebas del sistema, el sistema funciona bien en los dominios especificados, sin embargo si se introdujese un texto que se correspondiese a otro dominio el sistema no devolvería los resultados esperados. Lo único que sí que haría sería eliminar las palabras vacías, artículos y preposiciones. La solución para esto es ir añadiendo los dominios que se consideren necesarios. Es decir, se puede concluir que el sistema diseñado, necesita la existencia de dominios específicos para una categorización correcta de los

términos. Esto hace que el sistema sea muy configurable y que pudiese variar en función de las necesidades del cliente

Al partir con la premisa que los usuarios van a introducir texto correctamente escrito pueden surgir problemas al procesar el texto debido a que no hay un corrector de ortografía y por tanto el sistema no daría los resultados esperados.

7.1 Líneas de trabajo futuras

Como se ha visto en el apartado anterior y a lo largo de esta memoria, hay varios aspectos que se podrían mejorar del proyecto. A continuación se muestran posibles mejoras para el proyecto.

En primer lugar para mejorar el rendimiento del sistema sería introducir un corrector ortográfico, es decir, garantizar que todos los términos que entren al sistema estén escritos correctamente.

Otro de los puntos a mejorar sería el reconocimiento de nombres después de los artículos, debido a que es una posible fuente de errores porque en el caso que el nombre venga precedido de un adjetivo es éste el que se etiqueta como nombre en vez de adjetivo que sería lo correcto.

Como se ha visto en la parte del diseño, por cuestiones de facilitar el desarrollo, la primera letra de la oración se transforma siempre en minúscula, por tanto uno de los puntos a mejorar sería eliminar esa premisa y permitir que la primera letra pudiese ir en mayúscula. En un principio no se ve como una tarea fácil ya que puede ser que el comienzo de oración sea un nombre propio o bien que la primera palabra se encuentre con su primera letra en mayúscula debido a que es principio de oración, con lo cual el sistema no podría caracterizar si se trata de un nombre propio.

Una de las líneas de trabajo futuras puede ser seguir desarrollando diferentes dominios para que el sistema sea capaz de reconocer otro tipo de consultas.

Uno de los puntos a desarrollar puede ser convertir el sistema en multilingüe, por ejemplo se puede dar soporte al inglés creando listas de términos que estuviesen en inglés y adaptando las reglas a la gramática de dicho idioma.

Al tener una salida como XML las oportunidades de seguir desarrollando el proyecto son muy amplias, por ejemplo, se puede adaptar al sistema para convertirlo en un sistema de respuesta automática. Un sistema de respuesta automática tiene que ser capaz de "entender" preguntas en lenguaje natural y devolver una única respuesta correcta al usuario. Por ejemplo, si el usuario buscase una reserva de un viaje, el sistema podría directamente devolver la mejor opción como respuesta directa.

Para poder adaptar el sistema a un sistema de respuesta automática se necesita un lugar donde encontrar las respuestas a las preguntas introducidas. Por lo general, estas respuestas del sistema se basan en una base de datos de información semántica, pudiendo por tanto dar contestación a preguntas cuya información esté contenida en esta base de datos.

El análisis de la consulta de la versión actual, trabaja solamente sobre determinadas palabras, para seguir mejorando el sistema habría que añadir un módulo lematizador que extraiga la raíz de las palabras para obtener el significado sin importar ni género, ni número, ni la terminación verbal. Un ejemplo de esta utilidad podría ser la reducción de las listas de palabras ya que el sistema se quedaría con la raíz del término, la cual contiene el significado. Como ejemplo de un módulo de análisis morfosintáctico tenemos STILUS o Freeling [19] los cuales son capaces de realizar un análisis morfológico, sintáctico; y en el caso de STILUS, aporta detección de entidades, lo cual sería una aportación muy buena a la hora de detectar el objeto sobre el que se está preguntando.

8 Anexo

A continuación se muestran las diferentes listas de palabras usadas durante este proyecto.

Listas generales.

```
$arrEmptyWords = array_flip(array('busco','buscando','como','conocida','contengan','cosa','cual','cuales','cuál','cuanta','cuanto','cómo','denominado','deseo','documento','ella','ellas','ello','ellos','encontrar','encuentra','es','estar','esta','está','están','esté','estén','estoy','fue','fué','gustaría','ha','han','hacer','hay','información','le','localiza','más','me','mi','mis','muestra','muestrame','necesito','nosotros','otros','puede','pueden','puedo','que','quien','quién','quienes','quiero','quiero información del','quiero información de','qué','saber','se','sea','sean','ser','señor','señora','sitio web','solicito','solo','sólo','son','su','también','te','tendrá','tengo','tiene','tienen','todos','tu','va','van','vosotros','voy','yo','él'));

$arrStopWordsArt = array_flip(array('el','la','lo','los','las','un','una','uno','unos','unas'));

$arrStopWordsPrep = array_flip(array('a','al','ante','bajo','con','contra','de','del','desde','durante','en','entre','hacia','hasta','mediante','para','por','según','sin','so','sobre','tras','vía'));

$arrImp = array('y'=>'AND ',','=>'AND ','pero'=>'BUT ','sí'=>'YES ','e'=>'AND ','no' =>'NOT ','o'=>'OR ','u'=>'OR ','sin'=>'NOT ','ni'=>'AND NOT ');

$arrQuestion = array('cuanto cuesta'=>'precio','cuánto cuesta'=>'precio','cuántos' => 'cantidad','cuantos' => 'cantidad' ,'cuanto tarda'=> 'tiempo','cuánto se tarda'=>'tiempo','cuanto se tarda'=>'tiempo','cuánto tarda'=>'tiempo','cuánto vale' => 'precio','cuanto vale'=>'precio','quién' => 'persona','quiénes' => 'personas','quien' => 'persona', 'quienes' => 'personas','donde'=>'localización','dónde'=>'localización','cuándo'=>'fecha','cuando'=>'fecha');
```

Listas de dominio Universidad.

```
define('ES_SUSTITUTE', 'asignatura|profesor|profesora|departamento|universidad');
define('ES_TEACH', 'imparte|impartir|impartirá|da|enseña|profesor');
define('ES_REPLACE', 'asignatura de|curso de');
define('ES_UNIVERSITY_FILTER', 'da|dar|darán|dará|imparte|enseña|impartirá');
```

Listas de dominio Viajes.

```
define('ES_DESTINATION', 'a|al|destino|en|hacia|hasta|para|visitar');
define('ES_ORIGIN', 'de|del|desde|entre|origen|salida');
define('ES_DEPT', 'sale|salir|salga|saliendo');
define('ES_ARRIVAL', 'aterrizan|aterriza|llega|llegue');
define('ES_COME_BACK', 'regreso|vuelta|volviendo|vuelva|vuelve|vuelvo');
define('ES_CURRENCY', 'euro|euros|dólar|dólares|dolar|dolares|€|$');
define('EXCEPTIONS', 'visitar a|ver a|nombre de');
define('TRAVEL_FILTER',
'días|dia|día|aterrizan|ciudad|cuesta|cueste|cuesten|aterriza|irme|ir|llegue|llega|llegar|llegará|lleguen|parten|partiendo|procedente|procede|viajes|viaje|vuelta|vuelva|vuelve|vuelvan|salen|sale|saliendo|salir|salga|salgan');
```

9 Bibliografía

Las referencias aquí incluidas están ordenadas por orden de citación en el texto.

[1] Ingeniería lingüística en Wikipedia. Disponible [Internet]:

http://es.wikipedia.org/wiki/Ingenier%C3%ADa_ling%C3%BC%C3%ADstica [28 febrero 2012]

[2] Definición de fonología. Disponible [Internet]:

http://definicion.de/fonologia/ [28 febrero 2012]

[3] Definición de fonema. Disponible [Internet]:

http://www.materialesdelengua.org/LENGUA/ortografia/nivelfonico.htm [28 febrero 2012]

[4] Morfología. Disponible [Internet]:

http://lengua.laguia2000.com/gramatica/morfologia-linguistica [28 febrero 2012]

[5] Sintaxis. Disponible [Internet]:

http://lengua.laguia2000.com/gramatica/la-sintaxis[5 marzo 2012]

[6] Semántica del lenguaje español. Disponible [Internet]:

http://www.profesorenlinea.cl/castellano/Semantica1.htm [7 marzo 2012]

[7] Definición de hiperónimo. Disponible [Internet]:

http://www.ejemplode.com/12-clases_de_espanol/1872-ejemplo_de_hiperonimo.html [8 marzo 2012]

[8] Definición de hipónimo. Disponible [Internet]:

http://es.wikipedia.org/wiki/Hip%C3%B3nimo [8 marzo 2012]

[9] Pragmática en el lenguaje español. Disponible [Internet]:

http://www.monografias.com/trabajos55/generalidades-de-pragmatica/generalidades-de-pragmatica.shtml [15 marzo 2012]

[10] Procesamiento Lenguaje Natural. Disponible [Internet]:

http://www.cicling.org/ampln/NLP.htm [20 marzo 2012]

[11] STILUS Aplicación informática de Ingeniería Lingüística. Disponible [Internet]:

http://mystilus.com/MorphosyntacticAnalyzer [21 marzo 2012]

[12] XML. Monografías. Disponible [Internet]:

http://www.monografias.com/trabajos7/xml/xml.shtml [28 noviembre 2011]

[13] World wide web Consortium. Disponible [Internet]:

http://www.w3c.es [22 marzo 2012]

[14] Tutorial para aprender a usar XML. Disponible [Internet]:

http://www.w3schools.com/ [29 noviembre 2012]

[15] Sitio oficial de PHP. Disponible [Internet]:

http://www.php.net/ [2 noviembre 2011]

[16] Sitio oficial de Wamp server. Disponible [Internet]:

http://www.wampserver.com/ [15 noviembre 2011]

[17] Sitio oficial del servidor apache. Disponible [Internet]:

http://www.apache.org/ [23 marzo 2012]

[18] Sitio oficial de MySQL. Disponible [Internet]:

http://www.mysql.com/ [23 marzo 2012]

[19] Página principal de Freeling. Disponible [Internet]:

http://nlp.lsi.upc.edu/freeling/ [30 marzo 2012]