
Mejora de la Comprensión Lectora mediante Analogías para la Inclusión



Trabajo de Fin de Grado
Curso 2018–2019

Autor

Irene Martín Berlanga
Pablo García Hernández

Director

Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo

Grado en Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid

Mejora de la Comprensión Lectora mediante Analogías para la Inclusión

**Trabajo de Fin de Grado en Ingeniería de Software
Departamento de Ingeniería de Software e Inteligencia
Artificial**

Autor

**Irene Martín Berlanga
Pablo García Hernández**

Director

**Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo**

Dirigida por el Doctor

**Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo**

**Grado en Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid**

4 de febrero de 2019

Autorización de difusión

Los abajo firmantes, matriculados en el Grado de Ingeniería de Software de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo de Fin de Grado: “Mejora de la Comprensión Lectora mediante Analogías para la Inclusión”, realizado durante el curso académico 2018-2019 bajo la dirección de Virginia Francisco Gilmartín y Gonzalo Rubén Mendez Pozo en el Departamento de Ingeniería de Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Nombre Del Alumno

4 de febrero de 2019

Dedicatoria

Texto de la dedicatoria...

Agradecimientos

Texto de los agradecimientos

Resumen

Resumen en español del trabajo

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Abstract in English.

Keywords

10 keywords max., separated by commas.

Índice

1. Introduction	1
1. Introducción	3
1.1. Motivación	3
1.2. Objetivos	4
1.3. Estructura de la memoria	5
2. Estado de la Cuestión	7
2.1. Lectura Fácil	7
2.2. Figuras retóricas	9
2.3. Servicios Web	10
2.3.1. Arquitectura Servicios Web	11
2.3.2. Ventajas de los Servicios Web	12
2.3.3. Desventajas de los Servicios Web	13
2.4. Procesamiento del Lenguaje Natural	13
2.4.1. ConceptNet	15
2.4.2. Thesaurus	17
2.4.3. Thesaurus Rex	19
2.4.4. Metaphor Magnet	20
2.4.5. Wordnet	21
2.4.6. EuroWordnet	22
2.5. Conclusiones	22
3. Trabajo Realizado	27
3.1. Trabajo realizado por Irene	27
3.2. Trabajo realizado por Pablo	28
4. Conclusiones y Trabajo Futuro	31

4. Conclusions and Future Work	33
A. Título	35
B. Título	37
Bibliografía	39

Índice de figuras

2.1. Logo Lectura Fácil	8
2.2. Estructura de un mensaje SOAP	12
2.3. Ejemplo Red Semántica	14
2.4. Resultados de ConcepNet para la palabra chaqueta	15
2.5. Resultados búsqueda Thesaurus Rex con la palabra <i>house</i>	20
2.6. Resultados búsqueda Metaphor Magnet con la palabra <i>house</i>	22
2.7. Ejemplos formato XML Metaphor Magnet	23
2.8. Resultados búsqueda Wordnet con la palabra <i>house</i>	24
2.9. Resultados búsqueda EuroWordnet con la palabra <i>casa</i>	25

Índice de tablas

Chapter 1

Introduction

Introduction to the subject area.

Capítulo 1

Introducción

Actualmente, existen diversos colectivos con un entendimiento limitado del castellano, lo que convierte a sus integrantes en personas vulnerables, ya que se las puede engañar fácilmente. Aunque existen multitud de herramientas online para facilitar el entendimiento de un lenguaje determinado a personas que no lo hablan, como pueden ser los servicios de traducción instantánea, en los que al escribir un texto se traduce automáticamente a cualquier lengua que el usuario desee. No existen herramientas que permitan definir palabras difíciles utilizando comparaciones o términos más sencillos para facilitar su comprensión. Por este motivo es importante ofrecer un servicio accesible a todo el mundo que pueda definir palabras de una manera clara y sencilla.

1.1. Motivación

En nuestra sociedad actual, existen ciertos colectivos como pueden ser las personas con algún tipo de trastorno cognitivo, inmigrantes, ancianos, analfabetos funcionales, niños, etc... que tienen dificultad para aprender conceptos complejos o no tan complejos. Existen multitud de palabras cuyo significado es bastante complicado de explicar de una manera sencilla, por lo que una solución para que cualquier persona lo pueda comprender es hacer uso de metáforas o analogías. De esta forma (como se verá más adelante), se puede asimilar el concepto de una manera más rápida y sencilla.

La dificultad para entender conceptos complejos supone una serie de limitaciones en la vida cotidiana, en la forma de relacionarse con otros individuos, en la vida profesional e incluso la vida personal. Por ejemplo, suponemos que un inmigrante necesita realizar gestiones para legalizar su estancia o firmar documentación. Si no tiene una buena comprensión del lenguaje no sabe las consecuencias de firmar los documentos porque no los

entiende. Otro ejemplo sería cuando una persona firma un contrato, como por ejemplo una póliza de seguros o un contrato de trabajo, en el cual añaden ciertas cláusulas que son incomprensibles.

Para ayudar principalmente a estas personas a que puedan entender el significado de cualquier palabra, y de esta forma superar algunas de sus limitaciones, vamos a desarrollar una aplicación que permita definir palabras complejas mediante comparaciones con otras más fáciles ya conocidas por ellos. Por ejemplo, si queremos explicar una palabra compleja como puede ser piraña, podemos describirla utilizando conceptos más simples de la siguiente manera: *“Una piraña nada como un pez y es agresiva como un león”*. Mediante esta comparación, alguien que desconozca completamente el significado de *piraña*, puede hacerse una idea muy aproximada de lo que es.

1.2. Objetivos

El objetivo principal es crear una aplicación que dada una palabra compleja para el usuario obtenga una definición clara y sencilla mediante símiles, analogías o metáforas. Esto se puede ver claramente en el ejemplo de la piraña del apartado anterior, ya que, de esa manera cualquier persona que no sepa lo que es una piraña pueda hacerse una idea de cómo es dicho animal y asimilar el nuevo concepto. Además, se utilizarán técnicas centradas en el usuario para diseñar una interfaz lo más usable posible y que así el usuario tenga una experiencia de uso satisfactoria. Por último, el producto se diseñará para que se encuentre al alcance de todas aquellas personas que lo necesiten. Para que todo ello ocurra, los principales objetivos tecnológicos a alcanzar son:

- La aplicación estará construida con servicios web que la doten de funcionalidad.
- Los servicios web desarrollados estarán disponibles en una API pública para que todo el mundo pueda utilizarlos.
- La aplicación se construirá de manera incremental, añadiéndole valor al producto poco a poco.
- Hacer un desarrollo que esté centrado en el usuario, que sea lo más amigable posible y que solucione problemas reales de una forma usable.
- La aplicación esté formada por distintos módulos independientes cada uno con una funcionalidad específica.

Por último, no se deben de olvidar los objetivos académicos de este trabajo: poner en práctica los conocimientos adquiridos durante el Grado y ampliar nuestros conocimientos gracias a la utilización de herramientas, lenguajes y metodologías nuevas.

Alcanzando los objetivos anteriormente descritos, se conseguirá obtener un producto de calidad, con una gran utilidad tanto social como académica, que puede ayudar a mucha gente a aprender ciertos conceptos de nuestro idioma de una manera más sencilla.

1.3. Estructura de la memoria

En el **capítulo dos** se presenta el Estado de la Cuestión, en el que se explicará que es la Lectura Fácil y como se aplica y se introducirán los conceptos de Procesamiento del Lenguaje Natural (PLN) y algunas herramientas que sirven para PLN, además se hablará de figuras retóricas y servicios web, en especial qué son, su arquitectura y las ventajas y desventajas de su uso.

En el **capítulo tres** se describe el trabajo realizado por cada uno de los autores.

Capítulo 2

Estado de la Cuestión

Actualmente, los servicios web son una tecnología utilizada a diario por la mayoría de la población ya sea para la traducción de texto o la consulta del tiempo.

Debido a esto, para el desarrollo de la aplicación utilizaremos esta tecnología. Además, las definiciones generadas deben de ser símiles, metáforas o analogías ya que mediante estas figuras retóricas, se puede comparar el término que se quiere definir con otro concepto más cotidiano para facilitar su comprensión. Estas definiciones deben de estar en un formato que sea comprensible por el mayor número de gente posible para ello se empleará una adaptación llamada “lectura fácil” que tiene como objetivo que los texto estén en un formato fácil de leer. Para crear estas definiciones se utilizará Inteligencia Artificial para buscar términos relacionados o sinónimos más sencillos de la palabra que se quiere definir, concretamente la rama del Procesamiento del Lenguaje Natural.

2.1. Lectura Fácil

Se llama lectura fácil a aquellos contenidos que han sido resumidos y reescritos con lenguaje sencillo y claro, de forma que puedan ser entendidos por personas con discapacidad cognitiva o discapacidad intelectual. Es decir, es la adaptación de textos, ilustraciones y maquetaciones que permite una mejor lectura y comprensión. Este trabajo se va a centrar en la lectura fácil aplicada a textos.

La lectura fácil surgió en Suecia en el año 1968, donde se editó el primer libro en la Agencia de Educación en el marco de un proyecto experimental. A continuación, en 1976, se creó en el Ministerio de Justicia un grupo de trabajo para conseguir textos legales más claros. En 1984 nació el primer periódico en lectura fácil, titulado "8 páginas", que tres años más tarde, en 1987, se



Figura 2.1: Logo Lectura Fácil

publicó de forma permanente en papel hasta que empezó a editarse en la web. En el año 2013, en México se produce la primera sentencia judicial en lectura fácil¹. En la actualidad, podemos distinguir los documentos en lectura fácil gracias al logo de la Figura 2.1.

Los documentos escritos en Lectura Fácil (Wikipedia, 2018a) son documentos de todo tipo que siguen las directrices internacionales de la IFLA² y de la Inclusion Europe³ en cuanto al contenido y la forma. Algunas pautas a seguir para escribir correctamente un texto en Lectura Fácil son (Óscar García Muñoz, 2012):

- Evitar mayúsculas fuera de la norma, es decir, escribir en mayúsculas sólo cuando lo dicten las reglas ortográficas, como por ejemplo, después de un punto o la primera letra de los nombres propios.
- Deben evitarse el punto y seguido, el punto y coma y los puntos suspensivos. El punto y aparte hará la función del punto y seguido.
- Evitar corchetes y signos ortográficos poco habituales, como por ejemplo: %, & y /.
- Evitar frases superiores a 60 caracteres y utilizar oraciones simples. Por ejemplo, la oración *Caperucita ha ido a casa de su abuela y ha desayunado con ella* es mejor dividirla en dos oraciones simples: *Caperucita ha ido a casa de su abuela* y *Caperucita ha desayunado con ella*.
- Evitar tiempos verbales como: futuro, subjuntivo, condicional y formas compuestas.
- Utilizar palabras cortas y de sílabas poco complejas. Por ejemplo: casa, gato, comer o mano.

¹<https://dilofacil.wordpress.com/2013/12/04/el-origen-de-la-lectura-facil/>

²International Federation of Library Associations and Institutions

³Una asociación de personas con discapacidad intelectual y sus familias en Europa

- Evitar abreviaturas, acrónimos y siglas.
- Alinear el texto a la izquierda.
- Incluir imágenes y pictogramas a la izquierda y su texto vinculado a la derecha.
- Evitar la saturación de texto e imágenes.
- Utilizar uno o dos tipos de letra como mucho.
- Tamaño de letra entre 12 y 16 puntos.
- Si el documento está paginado, incluir la paginación claramente y reforzar el mensaje de que la información continúa en la página siguiente.

Debemos también hacer hincapié en la distinción entre palabras fáciles y complejas, puesto que son de gran importancia para la lectura fácil. Las palabras complejas son aquellas que no se utilizan a menudo en nuestra sociedad, como por ejemplo: meliflúo o inefable. Es por ello que este tipo de palabras deben estar totalmente descartadas en la lectura fácil, y en su lugar debemos introducir palabras fáciles, que son aquellas que se utilizan asiduamente. La RAE dispone de un documento con las mil palabras más usadas⁴.

2.2. Figuras retóricas

Las figuras literarias (o retóricas) son formas no convencionales de utilizar las palabras, de manera que, aunque se emplean con sus acepciones habituales, se acompañan de algunas particularidades fónicas, gramaticales o semánticas, que las alejan de ese uso habitual, por lo que terminan por resultar especialmente expresivas. La metáfora, el símil y la analogía se basan en la comparación de dos conceptos: de origen (o tenor), que es el término literal (al que la metáfora se refiere) y el de destino (o vehículo), que es el término figurado. La relación que hay entre el tenor y el vehículo se denomina fundamento. Por ejemplo, en la metáfora *Tus ojos son dos luceros*, *ojos* es el tenor, *luceros* es el vehículo y el fundamento es la belleza de los ojos (Galiana y Casas, 1994).

En este trabajo vamos a trabajar con tres tipos de figuras retóricas (Calleja, 2017):

- Metáfora: Utiliza el desplazamiento de características similares entre dos conceptos con fines estéticos o retóricos. Por ejemplo, cuando una tarea es muy fácil de realizar, se dice que es un regalo.

⁴<http://corpus.rae.es/lfrecuencias.html>

- **Símil:** Realiza una comparación entre dos términos usando conectores (por ejemplo, como, cual, que, o verbos). Por ejemplo, cuando nos referimos a una persona que es muy corpulenta, se dice: “es como un oso”, ya que los osos son muy grandes.
- **Analogía:** Es la comparación entre varios conceptos, indicando las características que permiten dicha relación. En la retórica, una analogía es una comparación textual que resalta alguna de las similitudes semánticas entre los conceptos protagonistas de dicha comparación. Por ejemplo: *Sus ojos son azules como el mar*.

2.3. Servicios Web

Para definir el concepto de servicio Web de la forma más simple posible, se podría decir que es una tecnología que utiliza un conjunto de protocolos para intercambiar datos entre aplicaciones, sin importar el lenguaje de programación en el cual estén programadas o ejecutadas en cualquier tipo de plataforma (Wikipedia, 2018c). Según el W3C (*World Wide Web Consortium*)⁵, un servicio web es un sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable.

Las principales características de un servicio web son (Wikipedia, 2017a):

- **Accesible a través de la Web.** Para ello debe utilizar protocolos de transporte estándares como HTTP, y codificar los mensajes en un lenguaje estándar que pueda ser accesible por cualquier cliente que quiera utilizar el servicio.
- **Contiene una descripción de sí mismo.** De esta forma, una aplicación web podrá saber cual es la función de un determinado Servicio Web, y cuál es su interfaz, de manera que pueda ser utilizado de forma automática por cualquier aplicación, sin la intervención del usuario.
- **Debe ser localizado.** Deberemos tener algún mecanismo que nos permita encontrar un Servicio Web que realice una determinada función. De esta forma tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente el usuario.

Por otro lado, los servicios web pueden definirse tanto a nivel conceptual como a nivel técnico, es por ello que mediante este último podemos diferenciar dos tipos distintos de servicio web (Wikipedia, 2017a):

- **Servicios web SOAP (Simple Object Access Protocol):** es un protocolo basado en XML para el intercambio de información entre ordenadores.

⁵<https://www.w3.org/>

Normalmente utilizaremos SOAP para conectarnos a un servicio e invocar métodos remotos ⁶. Los mensajes SOAP tienen el siguiente formato representado en la figura 2.2:

- **<Envelope>**: Elemento raíz de cada mensaje SOAP y contiene dos elementos: **<Header>** que es opcional y **<Body>** que es obligatorio, ambos elementos los describiremos a continuación.
 - **<Header>**: Es un elemento que se utiliza para indicar información acerca de los mensajes SOAP.
 - **<Body>**: Elemento que contiene información dirigida al destinatario del mensaje.
 - **<Fault>**: Elemento en el que se notifican los errores.
-
- **Servicios Web RESTful**: es un protocolo que suele integrar mejor con HTTP que los servicios basado en SOAP, ya que no requieren mensajes XML. Cada petición del cliente debe contener toda la información necesaria para entender la petición, y no puede aprovecharse de ningún contexto almacenado en el servidor.

2.3.1. Arquitectura Servicios Web

Hay que distinguir tres partes fundamentales en los servicios web (Wikipedia, 2017a):

- **El proveedor**: Es la aplicación que implementa el servicio y lo hace accesible desde Internet.
- **El solicitante**: Cualquier cliente que necesite utilizar el servicio web.
- **El publicador**: Se refiere al repositorio centralizado en el que se encuentra la información de la funcionalidad disponible y como se utiliza.

Por otro lado, los servicios web se componen de varias capas ⁷:

- **Service Discovery**: Responsable de centralizar los servicios web en un directorio común, de esta forma es más sencillo buscar y publicar.
- **Service Description**: Como ya hemos comentado con anterioridad, los servicios web se pueden definir así mismos, por lo que una vez que los localicemos el Service Description nos dará la información para saber que operaciones soporta y como activarlo.

⁶<https://www.ibm.com/support/knowledgecenter/es>

⁷<https://diego.com.es/introduccion-a-los-web-services>

```

<?xml version='1.0' Encoding='UTF-8' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      <n:name>Fred Bloggs</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2007-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2007-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference></p:seatPreference>
      </p:return>
    </p:itinerary>
  </env:Body>
</env:Envelope>

```

Figura 2.2: Estructura de un mensaje SOAP

- **Service Invocatio:** Invocar a un Web Service implica pasar mensajes entre el cliente y el servidor. Por ejemplo, si utilizamos SOAP (Simple Object Access Protocol), el Service Invocation especifica cómo deberíamos formatear los mensajes request para el servidor, y cómo el servidor debería formatear sus mensajes de respuesta.
- **Transport:** Todos los mensajes han de ser transmitidos de alguna forma entre el servidor y el cliente. El protocolo elegido para ello es HTTP ((HyperText Transfer Protocol)).

2.3.2. Ventajas de los Servicios Web

Las principales ventajas del uso de los servicios web son las siguientes (Lebrún, 2005):

- Permiten la integración “justo-a-tiempo”: Esto significa que los solicitantes, los proveedores y los agentes actúan en conjunto para crear sistemas que son auto-configurables, adaptativos y robustos.

- Reducen la complejidad por medio del encapsulamiento: Un solicitante de servicio no sabe cómo fue implementado el servicio por parte del proveedor, y éste a su vez, no sabe cómo utiliza el cliente el servicio. Estos detalles se encapsulan en los solicitantes y proveedores. El encapsulamiento es crucial para reducir la complejidad.
- Promueven la interoperabilidad: La interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y el lenguaje.
- Abren la puerta a nuevas oportunidades de negocio: Los servicios web facilitan la interacción con socios de negocios, al poder compartir servicios internos con un alto grado de integración.
- Disminuyen el tiempo de desarrollo de las aplicaciones: Gracias a la filosofía de orientación a objetos que utilizan, el desarrollo se convierte más bien en una labor de composición.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

2.3.3. Desventajas de los Servicios Web

El uso de servicios web también tiene algunas desventajas ⁸:

- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear.
- Existe poca información de servicios web para algunos lenguajes de programación.
- Dependen de la disponibilidad de servidores y comunicaciones.

2.4. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN) es una rama de la Inteligencia Artificial que se encarga de investigar la manera de comunicar máquinas con personas mediante el uso del lenguaje natural (entendiendo como lenguaje natural el idioma usado con fines de comunicación por humanos, ya sea hablado o escrito, como pueden ser el español, el ruso o el inglés). El Procesamiento del Lenguaje Natural se ayuda de las redes semánticas, puesto que estas son una forma de representación del conocimiento en la que los conceptos que componen el mundo y sus relaciones se representan mediante un grafo. Se utilizan para representar mapas conceptuales y mentales (Wikipedia,

⁸<http://fabioalfarocc.blogspot.com/2012/08/ventajas-y-desventajas-del-soap.html>

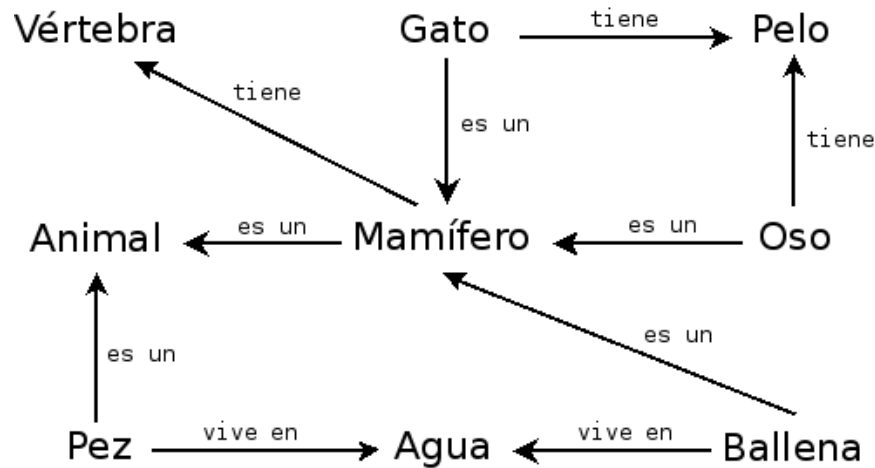


Figura 2.3: Ejemplo Red Semántica

2018b). Los nodos están representados por el elemento lingüístico, y la relación entre los nodos sería la arista. Podemos ver un ejemplo en la Figura 2.3, donde el nodo *Oso* representa un concepto, en este caso un sustantivo que identifica a un tipo de animal, y otro nodo *Pelo*. Su relación se ve representada por la arista con valor *tiene*, dando lugar a una característica de este animal: *Oso tiene pelo*.

Existen principalmente tres tipos de redes semánticas⁹:

- Redes de Marcos: los enlaces de unión de los nodos son parte del propio nodo, es decir, se encuentran organizados jerárquicamente, según un número de criterios estrictos, como por ejemplo la similitud entre nodos.
- Redes IS-A: los enlaces entre los nodos están etiquetados con una relación entre ambos. Es el tipo que habitualmente se utiliza junto con las Redes de Marcos. Por ejemplo, *la ballena vive en el agua*.
- Grafos Conceptuales: existen dos tipos de nodos: nodos de conceptos, los cuáles representan una entidad, un estado o un proceso y los nodos de relaciones, que indican como se relacionan los nodos de concepto. En este tipo de red semántica no existen enlaces entre los nodos con una etiqueta, sino que son los propios nodos los que tienen el significado. Por ejemplo.....

Para el trabajo que queremos realizar, existen varias aplicaciones web que actúan como redes semánticas y son capaces de procesar el Lenguaje Natural. A continuación, hablaremos de algunas de ellas.

⁹<http://elies.rediris.es/elies9/4-3-2.htm>

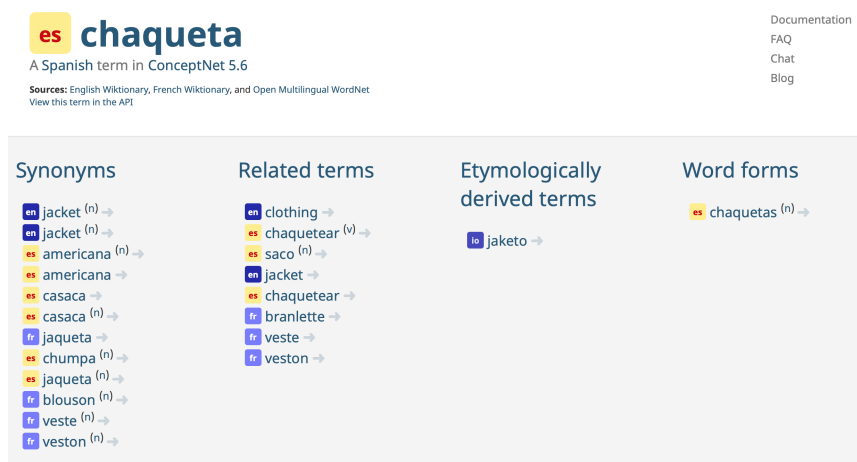


Figura 2.4: Resultados de ConcepNet para la palabra chaqueta

2.4.1. ConceptNet

Es una red semántica creada por el MIT (*Massachusetts Institute of Technology*) en 1999, diseñada para ayudar a los ordenadores a entender el significado de las palabras. Está disponible en múltiples idiomas, como el español, el inglés o el chino. ConceptNet dispone de una aplicación web¹⁰, donde seleccionas el idioma deseado y añades la palabra a buscar. En la Figura 2.4, podemos ver que si añadimos la palabra chaqueta nos devuelve sus sinónimos, términos relacionados, símbolos, etc... Por otro lado, ConcepNet dispone de un servicio web¹¹ que devuelve los resultados en formato JSON. Siguiendo con el mismo ejemplo anterior, podemos ver en la Figura 2.4.1 los resultados en dicho formato para la palabra chaqueta. Este consta de cuatro campos¹²:

- @context: URL enlazada a un archivo de información del JSON para comprender la API. También puede contener comentarios que pueden ser útiles para el usuario.
- @id: concepto que se ha buscado y su idioma. En nuestro caso, aparece de la siguiente manera: `/c/es/chaqueta`, donde *c* significa que es un concepto o término, *es* indica el lenguaje, en este caso, el español y por último *chaqueta* que es la palabra buscada.
- edges: representa una estructura de datos devueltos por Conceptnet compuesta por:

¹⁰<http://conceptnet.io/>

¹¹<http://api.conceptnet.io>

¹²principales <https://github.com/commonsense/conceptnet5/wiki/AP>

- @id: describe el tipo de relación que existe entre la palabra introducida y la devuelta, en nuestro caso `/a/[r/Synonym/,/c/es/chaqueta/n/,/c/es/americana/` nos indica que la palabra *americana* es un sinónimo de *chaqueta*.
 - @type: define el tipo del id, es decir, si es una relación (edge) o un término (nodo).
 - dataset: URI que representa el conjunto de datos creado.
 - end: nodo destino, que a su vez se compone de:
 - @id: coincide con la palabra del id anterior.
 - @type: define el tipo de id, como se ha explicado anteriormente.
 - label: frase más completa, donde adquiriera significado la palabra obtenida.
 - language: lenguaje en el que está la palabra devuelta de la consulta.
 - term: enlace a una versión mas general del propio término. Normalmente, suele coincidir con la URI.
 - license: aporta información sobre como debe usarse la información proporcionada por conceptnet.
 - rel: describe la relación que hay entre la palabra origen y destino, dentro del cual hay tres campos: @id, @type y label, descritos anteriormente.
 - sources: indica por qué ConceptNet guarda esa información, este campo como los anteriores, es un objeto que tiene su propio id y un campo @type, A parte, hay un campo *contributor*, en el que aparece la fuente por la que se ha obtenido ese resultado y por último un campo *process* indicando si la palabra se ha añadido mediante un proceso automático.
 - start: describe el nodo origen, es decir, la palabra que hemos introducido en ConceptNet para que haga la consulta, este campo esta compuesto por elementos ya descritos como son: @id, @type, label, language y term.
 - surfaceText: indica de que frase del lenguaje natural se han extraído los datos que estan guardados en conceptnet. En nuestro caso es nulo.
 - weight: indica la fiabilidad de la información guardada en conceptnet, siendo normal que su valor sea 1.0. Cuanto mayor sea este valor, más fiables serán.
- view: describe la longitud de la lista de paginación, es un objeto con un id propio, y además, aparecen los campos *firstPage* que tiene como valor un enlace a la primera pagina de los resultados obtenidos, y *nextPage* que tiene un enlace a la siguiente página de la lista.

JSON devuelto por la API de ConceptNet para la palabra chaqueta

```
{
  "@context": [
    "http://api.conceptnet.io/ld/conceptnet5.6/context.ld.json"
  ],
  "@id": "/c/es/chaqueta",
  "edges": [
    {
      "@id": "/a/[r/Synonym/,c/es/chaqueta/n/,c/es/americana/]",
      "@type": "Edge",
      "dataset": "/d/wiktionary/fr",
      "end": {
        "@id": "/c/es/americana",
        "@type": "Node",
        "label": "americana",
        "language": "es",
        "term": "/c/es/americana"
      },
      "license": "cc:by-sa/4.0",
      "rel": {
        "@id": "/r/Synonym",
        "@type": "Relation",
        "label": "Synonym"
      },
      "sources": [
        {
          "@id": "/and[/s/process/wikiparsec/1/,/s/resource/wiktionary/fr/]",
          "@type": "Source",
          "contributor": "/s/resource/wiktionary/fr",
          "process": "/s/process/wikiparsec/1"
        }
      ],
      "start": {
        "@id": "/c/es/chaqueta/n",
        "@type": "Node",
        "label": "chaqueta",
        "language": "es",
        "sense_label": "n",
        "term": "/c/es/chaqueta"
      },
      "surfaceText": null,
      "weight": 1.0
    },
    {
      "view": {
        "@id": "/c/es/chaqueta?offset=0&limit=20",
        "@type": "PartialCollectionView",
        "comment": "There are more results. Follow the 'nextPage' link for more.",
        "firstPage": "/c/es/chaqueta?offset=0&limit=20",
        "nextPage": "/c/es/chaqueta?offset=20&limit=20",
        "paginatedProperty": "edges"
      }
    }
  ]
}
```

2.4.2. Thesaurus

Es una aplicación web¹³ que se autodefine como principal diccionario de sinónimos de la web. Esta página ofrece la posibilidad de introducir una palabra para poder conocer sus sinónimos, pero solamente devuelve resultados en inglés. Aparte del listado de sinónimos, Thesaurus te dice que tipo de palabra es y una definición de la misma. Esta aplicación proporciona una API tipo RESTful¹⁴ que obtiene los sinónimos de una palabra y devuelve los resultados en formato XML o JSON. El contenido de la respuesta es una

¹³<https://www.thesaurus.com/>

¹⁴<http://thesaurus.altervista.org/>

lista y cada elemento de esta lista contiene un par de elementos: categoría y sinónimos. Este último a su vez contiene una lista de sinónimos separados por el carácter |. Podemos ver en la Figura 2.1 un ejemplo de como sería el resultado de una petición en formato XML y en la Figura 2.2 en formato JSON. Ambos son muy similares, por ejemplo en formato XML podemos ver que devuelve el tipo de categoría de las palabras, en este caso son sustantivos y a continuación aparecen los sinónimos. En caso de que alguna palabra sea un antónimo apareciera entre paréntesis al lado de la misma, como ocurre con la palabra *war*. Por otro lado, el formato JSON devuelve dentro del campo *category* / *categoría* todos los sinónimos, y en caso de ser un antónimo aparecerá de la misma forma que en el formato XML.

Listing 2.1: Ejemplo de salida de Thesaurus en formato XML

```
<response>
  <list>
    <category>(noun)</category>
    <synonyms> order | war (antonym) </synonyms>
  </list>
  <list>
    <category>(noun)</category>
    <synonyms> harmony | concord | concordance </synonyms>
  </list>
  <list>
    <category>(noun)</category>
    <synonyms> public security | security </synonyms>
  </list>
  <list>
    <category>(noun)</category>
    <synonyms> tratado de paz | pacificacion | acuerdo | pact | accord </
      synonyms>
  </list>
```

Listing 2.2: Ejemplo de salida de Thesaurus en formato JSON

```
{
  "response":
  [
    {
      "list":
      {
        "category": "(noun)", "sinonimos": "order | war (antonym)"
      }
    },
    {
      "list":
      {
        "category": "(noun)", "sinonimos": "armonia | concordia | concordancia"
      }
    },
    {
      "list":
      {
        "categoria": "(sustantivo)", "sinonimos": "seguridad publica |
        seguridad"
      }
    },
    {
      "list":
      {
        "categoria": "(sustantivo)", "sinonimos": "tratado de paz |
        pacificacion | tratado | pacto | acuerdo"
      }
    }
  ]
}
```

2.4.3. Thesaurus Rex

Thesaurus Rex¹⁵ es una red semántica. La aplicación permite introducir una palabra y devuelve palabras relacionados, o bien puedes introducir un par de palabras (separadas por el operador &) y devolverá las categorías que comparten ambos términos. Thesaurus Rex solo admite palabras en inglés. En la Figura 2.5 podemos ver los resultados para la palabra *house*. La aplicación devuelve estos resultados en formato XML, y este como se puede ver en la Figura 2.3 los divide en distintos campos: *Categories*, *Modifiers* y *CategoryHeads*. Como se puede ver todos los resultados tienen un peso asignado, esto significa que cuanto mayor sea el peso mayor es la similitud con el concepto dado, y en la página aparecerá dicha palabra en un tamaño superior al resto. Los campos que se encuentran dentro del apartado *categories* son los resultados de mayor grano fino que Thesaurus Rex ha encontrado, los que se encuentran dentro de *modifiers* son atributos del concepto a buscar y por último los que se encuentran en *categoryHeads* son las categorías más simples que se han para dicho concepto. Thesaurus Rex utiliza la Web Semántica para generar sus resultados, con lo cual la información disponible no es fija, sino que varía según los datos actuales de la web. La ventaja de utilizar esta herramienta es que se encuentra en continua actualización, pero el inconveniente es que en algunos casos la información puede resultar un poco extraña dado que se crea semiautomáticamente desde contenido de la web (Calleja, 2017).

Listing 2.3: Ejemplo formatos XML Thesaurus Rex

```
<MemberData>
  <Categories kw="house">
    <Category weight="91"> large:object </Category>
    <Category weight="307"> inanimate:object </Category>
    <Category weight="261"> everyday:object </Category>
    <Category weight="154"> sensitive:area </Category>
    <Category weight="318"> permanent:structure </Category>
    <Category weight="194"> permanent:construction </Category>
    <Category weight="148"> permanent:installation </Category>
    <Category weight="98"> fixed:object </Category>
  </Categories>

  <Modifiers kw="house">
    <Modifier weight="8"> recognizable </Modifier>
    <Modifier weight="9"> relevant </Modifier>
    <Modifier weight="863"> permanent </Modifier>
    <Modifier weight="15"> moderate </Modifier>
    <Modifier weight="477"> fixed </Modifier>
    <Modifier weight="5"> odd </Modifier>
    <Modifier weight="7"> archaeological </Modifier>
    <Modifier weight="5"> electrical </Modifier>
  </Modifiers>

  <CategoryHeads kw="house">
    <CategoryHead weight="6"> protection </CategoryHead>
    <CategoryHead weight="40"> obstruction </CategoryHead>
    <CategoryHead weight="5"> whole </CategoryHead>
    <CategoryHead weight="3320"> object </CategoryHead>
    <CategoryHead weight="2340"> structure </CategoryHead>
    <CategoryHead weight="98"> commodity </CategoryHead>
    <CategoryHead weight="713"> thing </CategoryHead>
    <CategoryHead weight="2"> theatre </CategoryHead>
  </CategoryHeads>
</MemberData>
```

¹⁵<http://ngrams.ucd.ie/therex3/>

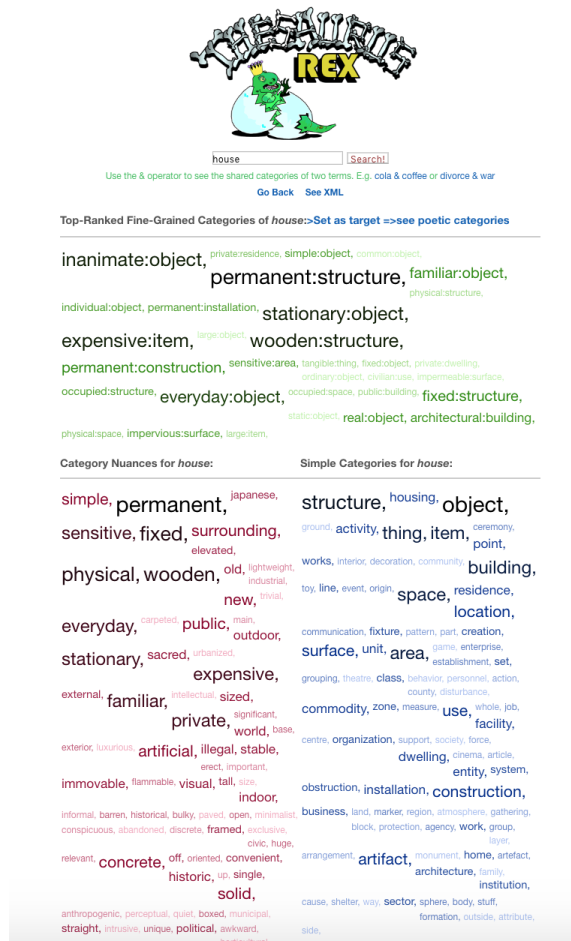


Figura 2.5: Resultados búsqueda Thesaurus Rex con la palabra *house*

```
</CategoryHeads>
</MemberData>
```

2.4.4. Metaphor Magnet

Aplicación web¹⁶, que dada una palabra crea metáforas. El objetivo (Veale y Li, 2012) de dicha aplicación es encontrar metáforas comunes en los n-gramas de Google. Se llama n-grama¹⁷ a una subsecuencia de n elementos consecutivos en una secuencia dada, es decir, contar la cantidad de veces que aparece una palabra, la cantidad de veces que aparecen dos palabras juntas, tres palabras, etc... Google utiliza estos mapeos para interpretar metáforas.

¹⁶<http://ngrams.ucd.ie/metaphor-magnet-acl/>

¹⁷<https://en.wikipedia.org/wiki/N-gram>

Esta aplicación está limitada, ya que como podemos observar en la Figura 2.6 introduciendo la palabra *house* solo está disponible para el inglés. Esta consulta devuelve un fichero XML como el expuesto en la Figura 2.4, que puede ser utilizado para otras aplicaciones de Procesamiento de Lenguaje Natural.

Listing 2.4: Ejemplo formatos XML Metaphor Magnet

```
<Metaphor>
  <Source Name="house">
    <Text> towering:mountain </Text>
    <Score> 88 </Score>
  </Source>
  <Source Name="house">
    <Text> protecting:home </Text>
    <Score> 86 </Score>
  </Source>
  <Source Name="house">
    <Text> tall:building </Text>
    <Score> 86 </Score>
  </Source>
  <Source Name="house">
    <Text> charming:castle </Text>
    <Score> 85 </Score>
  </Source>
  <Source Name="house">
    <Text> beautiful:tree </Text>
    <Score> 84 </Score>
  </Source>
  <Source Name="house">
    <Text> charming:mansion </Text>
    <Score> 83 </Score>
  </Source>
  <Source Name="house">
    <Text> strong:rock </Text>
    <Score> 80 </Score>
  </Source>
  <Source Name="house">
    <Text> strong:elephant </Text>
    <Score> 80 </Score>
  </Source>
</Metaphor>
```

2.4.5. Wordnet

Es una base de datos léxica solamente en inglés donde agrupa las palabras en función de su significado (University, 2010). Utiliza distintos tipos de palabras tales como sustantivos, verbos, adjetivos y adverbios ignorando preposiciones, determinantes y otras palabras funcionales. Agrupando estos tipos en conjuntos de sinónimos cognitivos o también llamados *synsets*. Estos, aparte contienen una breve definición y en muchas ocasiones oraciones cortas que explican el significado. En el caso de que la palabra tenga distintos significados se representará en tantos *synsets* como significados haya. Podemos ver un ejemplo en la figura 2.8 realizando una búsqueda con la palabra *house*. WordNet dispone de una API HTTP donde permite buscar información en la base de datos de WordNet mediante métodos GET y POST a través de la URL raíz */graph/v1*.

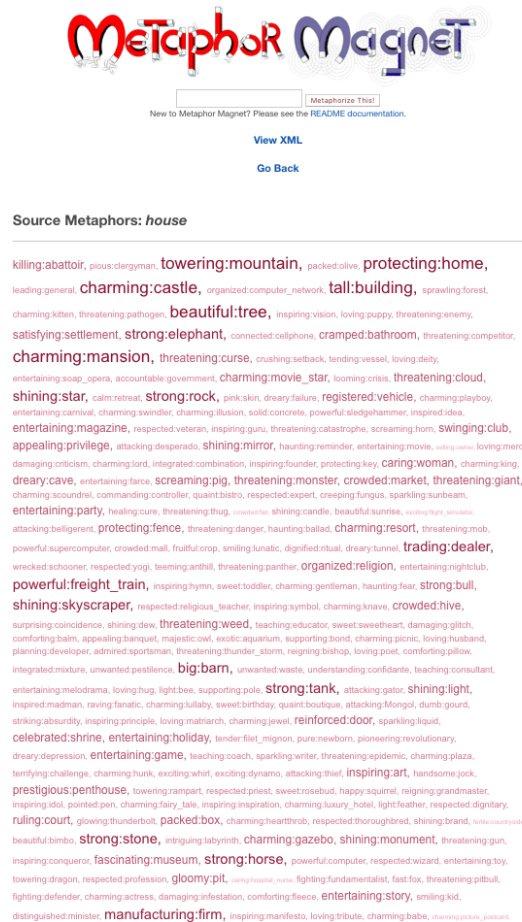


Figura 2.6: Resultados búsqueda Metaphor Magnet con la palabra *house*

2.4.6. EuroWordnet

Es una base de datos multilingüe que se basa en relaciones semánticas básicas entre palabras. Es muy parecido a WordNet pero en vez de usar *synsets* ellos lo llaman *WordNet*. Cada *WordNet* representa un sistema único de lexicalización y además estos están conectados con el Índice Inter-Lingual (ILI) (Wikipedia, 2017b). Podemos ver en la figura 2.9 un ejemplo de búsqueda con la palabra *casa*, donde la aplicación devuelve los sinónimos, una pequeña definición así como un ejemplo para que el usuario pueda entender mejor el concepto.

2.5. Conclusiones

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼<Metaphor>
  ▼<Source Name="house">
    <Text> towering:mountain </Text>
    <Score> 88 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> protecting:home </Text>
    <Score> 86 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> tall:building </Text>
    <Score> 86 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> charming:castle </Text>
    <Score> 85 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> beautiful:tree </Text>
    <Score> 84 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> charming:mansion </Text>
    <Score> 83 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> strong:rock </Text>
    <Score> 80 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> strong:elephant </Text>
    <Score> 80 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> shining:star </Text>
    <Score> 80 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> shining:skyscraper </Text>
    <Score> 79 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> towering:fortress </Text>
    <Score> 79 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> strong:tank </Text>
    <Score> 79 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> majestic:whale </Text>
    <Score> 79 </Score>
  </Source>
  ▼<Source Name="house">
    <Text> big:barn </Text>
    <Score> 77 </Score>
  </Source>

```

Figura 2.7: Ejemplos formato XML Metaphor Magnet

WordNet Search - 3.1
[- WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options:

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
 Display options for sense: (gloss) "an example sentence"

Noun

- [S:](#) (n) [home](#), [place](#) (where you live at a particular time) *"deliver the package to my home"; "he doesn't have a home to go to"; "your place or mine?"*
- [S:](#) (n) [dwelling](#), [home](#), [domicile](#), [abode](#), [habitation](#), [dwelling house](#) (housing that someone is living in) *"he built a modest dwelling near the pond"; "they raise money to provide homes for the homeless"*
- [S:](#) (n) [home](#) (the country or state or city where you live) *"Canadian tariffs enabled United States lumber companies to raise prices at home"; "his home is New Jersey"*
- [S:](#) (n) [home plate](#), [home base](#), [home](#), [plate](#) ((baseball) base consisting of a rubber slab where the batter stands; it must be touched by a base runner in order to score) *"he ruled that the runner failed to touch home"*
- [S:](#) (n) [base](#), [home](#) (the place where you are stationed and from which missions start and end)
- [S:](#) (n) [home](#) (place where something began and flourished) *"the United States is the home of basketball"*
- [S:](#) (n) [home](#) (an environment offering affection and security) *"home is where the heart is"; "he grew up in a good Christian home"; "there's no place like home"*
- [S:](#) (n) [family](#), [household](#), [house](#), [home](#), [menage](#) (a social unit living together) *"he moved his family to Virginia"; "It was a good Christian household"; "I waited until the whole house was asleep"; "the teacher asked how many people made up his home"; "the family refused to accept his will"*
- [S:](#) (n) [home](#), [nursing home](#), [rest home](#) (an institution where people are cared for) *"a home for the elderly"*

Verb

- [S:](#) (v) [home](#) (provide with, or send to, a home)
- [S:](#) (v) [home](#) (return home accurately from a long distance) *"homing pigeons"*

Adjective

- [S:](#) (adj) [home](#) (used of your own ground) *"a home game"*
- [S:](#) (adj) [home](#) (relating to or being where one lives or where one's roots are) *"my home town"*
- [S:](#) (adj) [home](#), [interior](#), [internal](#), [national](#) (inside the country) *"the British Home Office has broader responsibilities than the United States Department of the Interior"; "the nation's internal politics"*

Figura 2.8: Resultados búsqueda Wordnet con la palabra *house*

casa Look up ☒ Gloss ☐ English_3.0 ☐ Catalan_3.0
☐ Word ☐ Nouns ☐ Spanish_3.0 ☐ Score ☐ Basque_3.0 ☐ Portuguese_3.0
☐ near_synonym ☐ Spanish_3.0 ☐ Rels ☒ Spanish_3.0 ☐ Full ☐ Galician_3.0

Multilingual Central Repository (ILI 3.0) - [WikiMCR](#)

ili-30-02913152-n
buildings
town_planning
[Art_architecture_and_archaeology](#)
base concept
object_1 spa-30-02913152-n 298 edificio_1 inmueble_1 casa_1 una estructura que tiene un techo, paredes y se encuentra más o menos permanente en un solo lugar: había un edificio de tres pisos en la esquina; se trataba de un edificio imponente ;
artefact
Building=
Artifact=
Building=
Object=

ili-30-03544360-n
buildings
town_planning
[Art_architecture_and_archaeology](#)
base concept
building_1 spa-30-03544360-n 68 casa_2 una vivienda que sirve de residencia para una o más familias: él tiene una casa en Cape Cod;
artefact
Building+
Artifact=
Building=
Group+
Object=

Figura 2.9: Resultados búsqueda EuroWordnet con la palabra *casa*

Capítulo 3

Trabajo Realizado

En este capítulo vamos a describir que trabajo hemos hecho cada uno

3.1. Trabajo realizado por Irene

Primero investigué las bibliotecas que utilizaremos para el procesado de las palabras, al principio encontramos una biblioteca para el procesado de texto en Python, que es la nltk pero vimos que las etiquetas que ponía a las palabras no eran del todo correctas por lo que buscamos otra biblioteca y encontramos Spacy, con esta ya pudimos etiquetar bien todas las palabras diseñando un programa inicialmente en el Jupyter. A continuación, investigué que tecnologías utilizar para la realización del prototipo tecnológico, encontramos como entorno de desarrollo Pycharm y como framework Django. Una vez seleccionadas las tecnologías, investigamos como se utilizaban y nos pusimos a trabajar en el prototipo tecnológico.

Yo me encargué de conectar las vistas html con la lógica en Python, a continuación vimos como se implementaba un formulario y como se hacia una redirección a vista. Cuando supimos como se hacia todo esto, integramos el código desarrollado en Jupyter en nuestro servicio web finalizando el prototipo tecnológico.

En cuanto a la memoria me la dividí a partes iguales con Pablo, intentando que los dos toquemos todo, por lo que ambos redactamos tanto una parte de la introducción (en la que redacté la motivación) como el estado de la cuestión (yo hice el apartado de lectura fácil y Procesamiento del Lenguaje Natural).

La investigación de como funcionaba Conceptnet y su API la hicimos conjuntamente.

3.2. Trabajo realizado por Pablo

Al igual que mi compañera, lo primero que hicimos fue investigar como podíamos etiquetar las palabras, encontramos la librería nltk de Python para hacerlo, pero tras un primer intento nos dimos cuenta de que muchas palabras no estaban etiquetadas como deberían por lo que decidimos buscar alternativas, indagando un poco encontramos Spacy, la probamos y obtuvimos unos resultados mucho mejores que con nltk por lo que decidimos utilizar esta última (todo esto lo hicimos desde el Jupyter).

Cuando terminamos de etiquetar las palabras nos pusimos a investigar herramientas para el desarrollo del prototipo tecnológico y nos decantamos por utilizar Django como framework integrado en Pycharm, que es el entorno de desarrollo.

A continuación empezamos el desarrollo del prototipo tecnológico primero investigando como se utilizaban estas tecnologías(implementar formularios, hacer las redirecciones a vista...). Para finalizar migramos lo hecho desde Jupyter a nuestro servicio web.

Irene y yo nos dividimos la redacción de la memoria de tal manera que los dos hicimos tanto la parte de la introducción como del estado de la cuestión, de la introducción a mí me tocó la parte de los objetivos y del estado de la cuestión la parte de figuras retóricas y servicios web.

La investigación de como funcionaba Conceptnet y su API la hicimos de manera conjunta.

El diseño de la Interfaz debe cumplir las Ocho Reglas de Oro del diseño de interfaces:

- **Consistencia:** La funcionalidad debe ser similar a otras aplicaciones que el usuario está acostumbrado a utilizar. En cuanto a la interfaz debe tener los mismos colores, iconos, formas, botones, mensajes de aviso... Por ejemplo, si el usuario está acostumbrado a que el botón de eliminar o cancelar sea rojo, no debemos añadirle uno de color verde.
- **Usabilidad Universal:** Debemos tener en cuenta las necesidades de los distintos tipos de usuario, como por ejemplo, atajos de teclado para un usuario experto o filtros de color para usuarios con deficiencias visuales.
- **Retroalimentación activa:** Por cada acción debe existir una retroalimentación legible y razonable por parte de la aplicación. Por ejemplo, si el usuario quiere guardar los datos obtenidos de la búsqueda, la aplicación debe informarle de si han sido guardados o no.
- **Diálogos para conducir la finalización:** El usuario debe saber en que paso se encuentra en cada momento. Por ejemplo, en un proceso de

compra que conlleva varios pasos hasta la finalización de la misma, se le debe informar donde se encuentra y cuánto le queda para terminar.

- **Prevención de errores:** La interfaz debe ayudar al usuario a no cometer errores serios, y en caso de cometerlos se le debe dar una solución lo más clara y sencilla posible. Por ejemplo, deshabilitando opciones o indicando en un formulario el campo en el cual se ha producido el error sin perder la información ya escrita.
- **Deshacer acciones fácilmente:** Se debe dar al usuario la capacidad de poder deshacer o revertir acciones de una manera sencilla.
- **Sensación de control:** Hay que dar al usuario la sensación de que tiene en todo momento el control de la aplicación, añadiendo contenidos fáciles de encontrar y de esta forma no causarle ansiedad o frustración por utilizar nuestra aplicación.
- **Reducir la carga de memoria a corto plazo:** La interfaz debe ser lo más sencilla posible y con una jerarquía de información evidente, es decir, hay que minimizar la cantidad de elementos a memorizar por el usuario.

Capítulo 4

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Chapter 4

Conclusions and Future Work

Conclusions and future lines of work.

Apéndice	A
----------	----------

Título

Contenido del apéndice

Apéndice	B
----------	----------

Título

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el cerebro de manera que vino
a perder el juicio.*

Miguel de Cervantes Saavedra

CALLEJA, P. G. *Generación de recursos lingüísticos mediante la extracción de relaciones entre conceptos*. Trabajo de fin de máster, Universidad Complutense de Madrid, 2017.

GALIANA, A. y CASAS, J. *Introducción al análisis retórico: tropos, figuras y sintaxis del estilo*. Universidad de Santiago de Compostela, 1994.

ÓSCAR GARCÍA MUÑOZ. *Lectura Fácil: Métodos de redacción y evaluación*. Real Patronato sobre Discapacidad, Ministerio de Sanidad, Servicios Sociales e Igualdad, 2012.

LEBRÚN, C. A. V. Integración de herramientas de tecnologías de información como soporte en la adm del conocimiento. 2005.

UNIVERSITY, P. About wordnet. 2010.

VEALE, T. y LI, G. *Exploding the Creativity Myth: The Computational Foundations of Linguistic Creativity*. Bllomsbury Academic, 2012.

WIKIPEDIA. Definición soap. 2017a.

WIKIPEDIA. Eurowordnet — wikipedia, la enciclopedia libre. 2017b.

WIKIPEDIA. Lectura fácil — wikipedia, la enciclopedia libre. 2018a.

WIKIPEDIA. Red semántica — wikipedia, la enciclopedia libre. 2018b.

WIKIPEDIA. Servicio web — wikipedia, la enciclopedia libre. 2018c.

*–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

*–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

