
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 Méndez 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

23 de mayo 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

23 de mayo 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. Metodología de gestión del Proyecto	5
1.4. Estructura de la memoria	6
2. Estado de la Cuestión	9
2.1. Procesamiento del Lenguaje Natural	10
2.1.1. ConceptNet	12
2.1.2. Thesaurus	16
2.1.3. Thesaurus Rex	17
2.1.4. Metaphor Magnet	20
2.1.5. WordNet	22
2.2. Lectura Fácil	24
2.3. Figuras retóricas	26
2.4. Servicios Web	26
2.4.1. Tipos de Servicios Web	27
2.4.2. Arquitectura Servicios Web	28
2.4.3. Ventajas de los Servicios Web	29
2.4.4. Desventajas de los Servicios Web	30
3. Herramientas Utilizadas	31
3.1. Django	31
3.2. SpaCy	32
4. Propuesta solución: Aplicación Aprende Fácil	35
4.1. Diseño centrado en el usuario de los prototipos de la Aplicación	35

4.1.1. Primera Iteración: Iteración Competitiva	35
4.1.2. Segunda Iteración: Evaluación con Expertos	39
4.2. Arquitectura de Aprende Fácil	41
4.3. Diseño de la Base de datos de Aprende Fácil	42
4.4. Diseño del Backend de Aprende Fácil	45
4.4.1. Selección de la Red Semántica a utilizar	45
4.4.2. Servicio Web para la obtención de Sinónimos Fáciles .	48
4.4.3. Servicio Web para la obtención de hipónimos fáciles .	49
4.4.4. Servicio Web para la obtención de hiperónimos fáciles	50
4.4.5. Servicio Web para la obtención de metáforas	50
4.4.6. Servicio Web para la obtención de símiles	51
4.5. Diseño del Frontend de Aprende Fácil	51
4.6. Evaluación de Aprende Fácil	54
5. Trabajo Realizado	77
5.1. Trabajo realizado por Irene	77
5.2. Trabajo realizado por Pablo	78
6. Conclusiones y Trabajo Futuro	79
6. Conclusions and Future Work	81
A. Artículos Periodísticos Utilizados para el análisis cualitativo y cuantitativo	83
A.1. Artículo Tecnológico	83
A.2. Artículo Deportivo	85
A.3. Artículo Político	87
B. Título	89
Bibliografía	91

Índice de figuras

1.1.	Tablero de tareas al inicio del proyecto	6
2.1.	Ejemplo de Red Semántica	11
2.2.	Ejemplo de Red de Marco	11
2.3.	Ejemplo de Red de IS-A	12
2.4.	Ejemplo de Grafo Conceptual	12
2.5.	Resultados de ConcepNet para la palabra <i>chaqueta</i>	13
2.6.	Resultados de ConcepNet para la palabra polisémica <i>book</i> . .	16
2.7.	Resultados búsqueda Thesaurus Rex con la palabra <i>house</i> . .	18
2.8.	Resultados búsqueda Metaphor Magnet con la palabra <i>house</i> .	21
2.9.	Resultados de la búsqueda en EuroWordNet para la palabra <i>casa</i>	23
2.10.	Logo Lectura Fácil	24
3.1.	Ejemplo de clasificación de palabras	33
4.1.	Prototipo de Pablo mostrando resultado más común para la palabra vehículo	39
4.2.	Prototipo de Pablo mostrando resultado más común para la palabra vehículo y un ejemplo	40
4.3.	Prototipo de Pablo mostrando todos los resultados para la palabra vehículo	41
4.4.	Prototipo de Pablo mostrando todos los resultados para la palabra vehículo junto con pictogramas	42
4.5.	Prototipo de Irene mostrando resultado más común para la palabra vehículo	43
4.6.	Prototipo de Irene mostrando resultado más común para la palabra vehículo y una definición	44
4.7.	Prototipo de Irene mostrando todos los resultados para la palabra portero	45

4.8. Prototipo de Irene mostrando todos los resultados para la palabra portero junto con pictogramas	58
4.9. Prototipo mostrando resultado más común para la palabra portero	59
4.10. Prototipo Versión 1 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo separados	60
4.11. Prototipo Versión 2 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo a simple vista	61
4.12. Prototipo Versión 3 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo en un mismo botón	62
4.13. Prototipo mostrando todos los resultados para la palabra portero	63
4.14. Prototipo mostrando todos los resultados para la palabra portero incluyendo los pictogramas	64
4.15. Diagrama de la arquitectura del proyecto	65
4.16. Relación entre <i>Backend</i> y <i>Frontend</i>	65
4.17. Tablas utilizadas en la Base de Datos	66
4.18. Tabla de resultados de la prueba cuantitativa	67
4.19. Tabla de resultados de la prueba cualitativa	68
4.20. JSON devuelto al buscar los sinónimos fáciles de inmueble en el servicio web para la obtención de sinónimos fáciles	68
4.21. JSON devuelto al buscar los sinónimos fáciles de inmueble	69
4.22. JSON devuelto al buscar los hipónimos fáciles de inmueble	70
4.23. JSON devuelto al buscar los hiperónimos fáciles de inmueble	71
4.24. JSON devuelto al buscar las metáforas de la palabra inmueble	72
4.25. JSON devuelto al buscar los símiles de la palabra inmueble	73
4.26. Interfaz de Aprende Fácil sin realizar ninguna búsqueda	74
4.27. Interfaz de Aprende Fácil mostrando resultados	75

Índice de tablas

Índice de Listados

2.1.	JSON devuelto por la API de ConceptNet para la palabra chaqueta	15
2.2.	XML devuelto por Thesaurus para la palabra <i>peace</i>	17
2.3.	JSON devuelto por Thesaurus para la palabra <i>peace</i>	17
2.4.	XML devuelto por Thesaurus Rex para la palabra <i>house</i> . . .	19
2.5.	XML devuelto por Metaphor Magnet para la palabra <i>house</i> .	20
2.6.	Estructura de un mensaje SOAP	28

Chapter 1

Introduction

Introduction to the subject area.

Introducción

1.1. Motivación

Hoy en día, el español es la segunda lengua más hablada del mundo y actualmente más de 90.000 palabras forman el castellano. Se trata de una lengua con multitud de términos, y que dependiendo del contexto en el que se encuentren, pueden tener múltiples significados. Por ejemplo, la palabra gato puede hacer referencia a un animal o a una herramienta para elevar un coche. Si esto supone una complicación para cualquier persona, para ciertos colectivos afectados por algún trastorno cognitivo lo es aún mucho más, afectándoles en su vida cotidiana, profesional o personal. Por ejemplo, el simple hecho de leer un periódico es una acción bastante difícil para ellos ya que muchas palabras no saben lo que significan. Otro ejemplo podría ser leer un manual de instrucciones, donde se encuentran en la misma situación de no poder entender ciertos conceptos.

Una de las soluciones que se podrían pensar en un primer momento, es buscar su significado en un diccionario. Pero esto no les sirve, puesto que las definiciones que aparecen en muchos casos no utilizan términos o frases sencillas. Por ejemplo, la palabra computadora tiene la siguiente definición en el diccionario: *Máquina electrónica que, mediante determinados programas, permite almacenar y tratar información, y resolver problemas de diversa índole*¹. Esta definición puede ser bastante complicada de entender por una persona con discapacidad cognitiva ya que utiliza términos más técnicos y es una frase bastante larga, por lo que una solución que se ha pensado para poder solventar dicho problema, es ofrecer una definición que compare el concepto en cuestión con otros conceptos más sencillos ya conocidos. Para hacer estas comparaciones usaremos las figuras retóricas, más concretamente haremos uso de las metáforas, analogías y símiles. Por ejemplo, para la

¹<https://dle.rae.es/?id=A4hIGQC>

palabra computadora se podrían obtener los siguientes resultados:

- *Una computadora es un ordenador.* (Metáfora)
- *Una computadora es como una máquina.* (Analogía)
- *Una computadora es fuerte como una piedra.* (Símil)

Creemos que el uso de las figuras retóricas facilitan el entendimiento del concepto. En estas definiciones se utilizarán frases cortas y conceptos sencillos. Además, añadiremos pictogramas a los resultados para llegar a un colectivo más amplio.

Por tanto, lo que proponemos es crear una aplicación que dada una palabra compleja devuelva los resultados mediante comparaciones con conceptos más sencillos y haciendo uso de las figuras retóricas.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es crear una aplicación web basada en servicios web que dada una palabra compleja para el usuario devuelva una definición de dicha palabra comparándola con otras palabras más sencillas mediante símiles, analogías o metáforas. Para poder obtener estas definiciones, habrá que estudiar como obtener los términos relacionados de un concepto, así como distinguir de estos resultados cuales son palabras fáciles y difíciles. También habrá que saber que tipos de figuras retóricas existen y cuál utilizar según la relación entre el concepto inicial y los conceptos fáciles, de esta forma se podrá facilitar al usuario final un resultado claro y correcto.

La aplicación estará construida con servicios web que doten de funcionalidad a la aplicación y que sean reutilizables en otras aplicaciones, haciendo así que se puedan adaptar a las distintas necesidades de los usuarios finales. Los servicios web desarrollados estarán disponibles en una API pública para que todo el mundo pueda utilizarlos y puedan servir para que otros desarrolladores integren nuestros servicios en sus aplicaciones.

La aplicación se construirá de manera incremental, añadiendo valor al producto poco a poco. De este modo se podrán testear las distintas hipótesis de trabajo poco a poco y realizar modificaciones de una manera simple para así conseguir una aplicación que se adecúe a las necesidades de los usuarios.

El diseño de la interfaz estará centrado en el usuario, y para ello se debe obtener la mayor información posible sobre el usuario final. De esta forma se realiza un diseño basado en sus necesidades y se obtendrá una mayor satisfacción de este al utilizar la aplicación. Como dicho trabajo está enfocado para personas con discapacidad cognitiva, se debe realizar un diseño que se adapte aún más a sus necesidades y limitaciones.

Por último, no se deben olvidar los objetivos académicos de este trabajo, como pueden ser poner en práctica los conocimientos adquiridos durante el grado y ampliar nuestros conocimientos en distintas áreas.

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

1.3. Metodología de gestión del Proyecto

Desde el inicio del proyecto se ha buscado la eficiencia y la mejora continua, es por ello que se han tenido reuniones con los directores de este trabajo, cada dos o tres semanas. En estas reuniones se revisaba el trabajo realizado desde la última reunión, se buscaban soluciones a los problemas y dudas que pudieran surgir o plantearse y se fijaban las tareas a realizar hasta la próxima reunión. Por otro lado, ha habido comunicación constante con ambos directores vía email para consultar dudas y también ha habido tutorías para resolver los problemas que iban surgiendo.

En relación a la gestión de configuración se ha utilizado la plataforma de desarrollo online GitHub para llevar el control de versiones del proyecto.

Además, se ha hecho uso de un gestor de tareas que ha servido como radiador de información y así todos los integrantes del proyecto han podido conocer en cada momento el estado del proyecto y de cada una de sus tareas. Para ello, se ha elegido Trello, ya que dispone de una interfaz simple, amigable y que no lleva a confusión a la hora de crear nuevas tareas. Existen dos tipos de tareas en nuestro tablero: las relacionadas con código y las relacionadas con la memoria. Se ha realizado una distinción entre ambas, ya que la forma de cambiar su estado en el tablero varía significativamente. Para hacer esta distinción en las tareas se ha escrito la palabra CÓDIGO o la palabra MEMORIA según corresponda delante de la descripción de la misma. Nuestro tablero de tareas tiene tres columnas:

- TO DO: Tareas a realizar, desgranadas al mayor detalle posible e intentando que éstas sean lo más independientes las unas de las otras. De esta forma, se asegura que cada integrante del equipo trabaja en una tarea específica que no influye en el trabajo del otro compañero.
- En proceso: En el momento en el que un integrante del equipo se asigna una tarea, esta se pasa de la columna TO DO a la columna “En proceso”, lo que indica que se encuentra en proceso de realización y que ningún otro compañero puede ponerse a trabajar en ella.
- En revisión: En esta columna se encuentran las tareas terminadas pero no validadas. La validación depende del tipo de tarea: si la tarea es

de tipo código, la validación la deben hacer los miembros del equipo realizando pruebas para comprobar que realmente cumple su objetivo y no provoca ningún error, y si es de tipo memoria la deben hacer los directores.

- Hecho: En esta columna se encuentran las tareas ya validadas por los directores o por los integrantes del equipo en función del tipo de tarea.

En la Figura 1.1 se muestra el estado del tablero al inicio del proyecto.

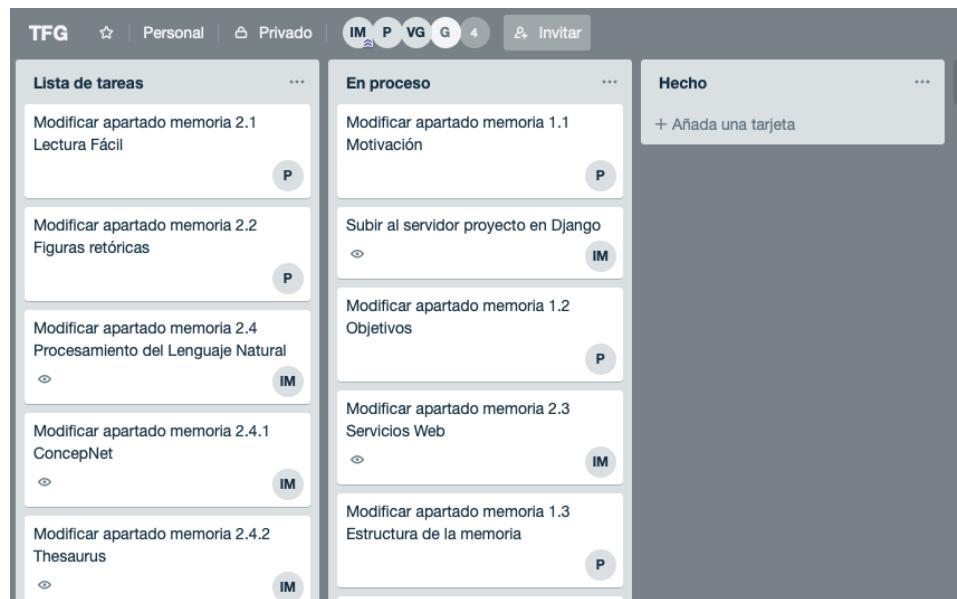


Figura 1.1: Tablero de tareas al inicio del proyecto

1.4. 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 explicarán las herramientas utilizadas para la creación de este trabajo, como pueden ser Django para el desarrollo de la aplicación y SpaCy para el etiquetado de palabras, donde se explicará que son ambas herramientas, para que se utilizan y sus características principales.

El **capítulo cuatro** se explicará todo lo relacionado con la implementación de la aplicación. Desde los primeros prototipos creados, las evaluaciones

con los expertos, la arquitectura de la misma y la explicación tanto de los servicios web implementados por los integrantes del trabajo para dotar de funcionalidad a la aplicación así como se ha conseguido la funcionalidad de la vista.

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

Capítulo 2

Estado de la Cuestión

Como se ha comentado en el anterior capítulo, el objetivo principal de este trabajo es construir una aplicación que ayude a personas con discapacidad cognitiva a poder entender palabras más complejas mediante conceptos más sencillos. Para poder conseguir cumplir dicho objetivo hay que seguir ciertos pasos.

En primer lugar, se deben buscar los conceptos relacionados con la palabra introducida por el usuario. Para ello, se ha utilizado una rama de la Inteligencia Artificial llamada Procesamiento del Lenguaje Natural (PLN). En la sección 2.1 se explicará en qué consiste y las distintas herramientas disponibles para la búsqueda de conceptos relacionados con una palabra específica.

Una vez obtenidas las palabras relacionadas con el concepto introducido, se debe hacer una distinción de cuales de las palabras relacionadas pertenecen al grupo de palabras fáciles y cuales al grupo de palabras difíciles, es decir, distinguir entre aquellas palabras que se utilizan asiduamente y las que no. Esta distinción es necesaria ya que uno de nuestros objetivos es trabajar únicamente con las palabras fáciles para facilitar la compresión del significado de un concepto devolviendo un resultado lo más sencillo posible. Dado que este trabajo está enfocado para personas con discapacidad cognitiva, el concepto de lectura fácil nos ayuda a comprender como debemos trabajar con las palabras fáciles y por ello en la sección 2.2 se explicará que es la lectura fácil y algunas pautas que se pueden seguir para escribir correctamente un texto con esta adaptación, así como la definición de palabras faciles y difíciles y los documentos que se utilizarán como referencia para las palabras fáciles.

Tras identificar las palabras fáciles relacionadas con el concepto introducido, se utilizarán figuras retóricas para comparar el concepto complejo con otros conceptos más sencillos que ayuden a entender el significado del concepto complejo introducido por el usuario. En la sección 2.3 se hablará de las figuras retóricas y se explicarán los tres tipos fundamentales que se van a utilizar para la realización de este trabajo.

Por último, vamos a hacer uso de servicios web para dotar a la aplicación de funcionalidad. Como se ha comentado en el anterior capítulo, uno de los objetivos tecnológicos de este trabajo es construir la aplicación con servicios web y en la sección 2.4 se explicará detalladamente que es un servicio web, los tipos que existen, sus características principales, su arquitectura y las ventajas de ser utilizados así como sus desventajas.

2.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN) es una rama de la Inteligencia Artificial que se encarga de la comunicación entre máquinas y 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).

Una de las tareas principales en el Procesamiento del Lenguaje Natural (PLN) es interpretar un texto escrito en lenguaje natural y entender su significado, entendiendo como significado la relación entre una palabra o una frase con el mundo. Para realizar dicha acción no solo es necesario el conocimiento del propio lenguaje en que está escrito el texto sino que también es necesario un conocimiento del mundo. Por tanto, uno de los grandes retos del PLN es la representación del conocimiento. Se deben de buscar técnicas que permitan representar conceptos y relaciones semánticas entre ellos.

Una de las principales técnicas de representación de conocimiento son las redes semánticas, en ellas los conceptos que componen el mundo y sus relaciones se representan mediante un grafo. Las redes semánticas se utilizan para representar mapas conceptuales y mentales (Quillian, 1968). Los nodos están representados por el elemento lingüístico, y la relación entre los nodos sería la arista. Se puede ver un ejemplo en la Figura 2.1, donde el nodo *Oso* representa un concepto, en este caso un sustantivo que identifica a un tipo de animal, y otro nodo *Pelo* el cual también es un sustantivo. La relación entre ambos 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 semánticas (Moreno Ortiz, 2000):

- 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. En la Figura 2.3, se muestra un ejemplo de Red de Marco donde por ejemplo el nodo ave tiene como características que vuela, que tiene plumas y que pone huevos, pero en cambio el nodo avestruz que es un tipo de ave, no puede volar. Por lo que los nodos de ave son las características principales de un ave, aunque no todas tienen por que

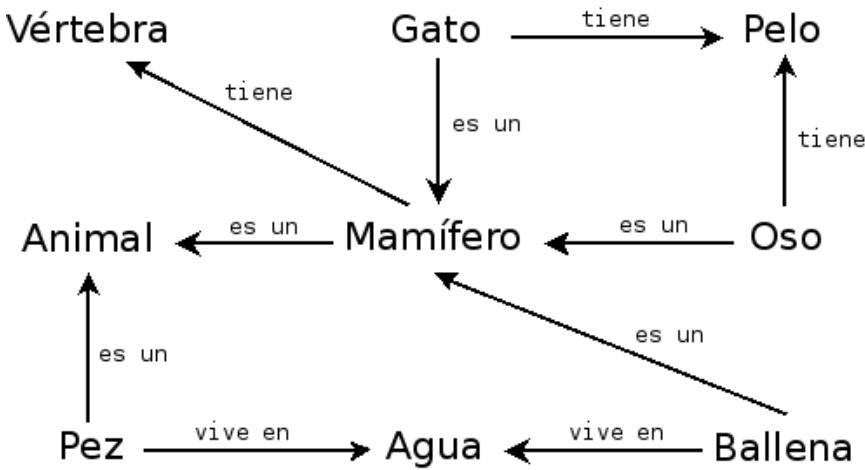


Figura 2.1: Ejemplo de Red Semántica

cumplirlo.

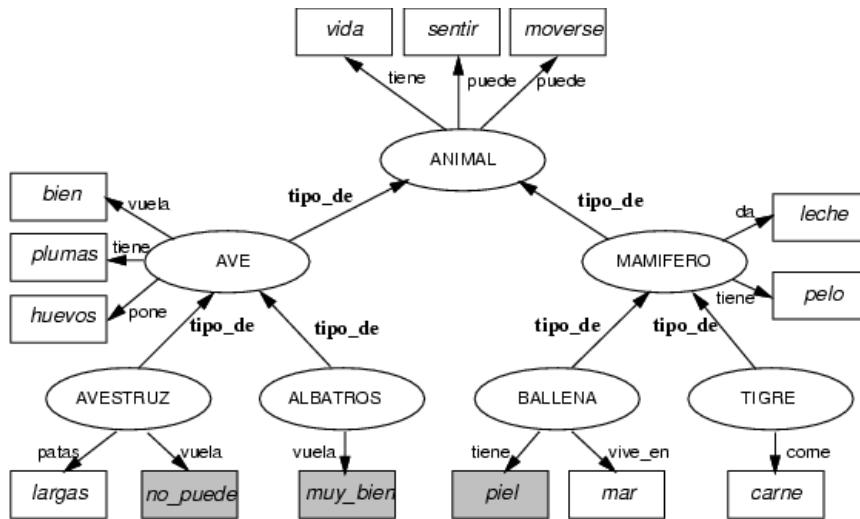


Figura 2.2: Ejemplo de Red de Marco

- 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. En la Figura 2.2 se muestra una red IS-A en la que se representa que: mujer y hombre son personas, y perro y gato son animales. Por último, tanto personas como animales son seres vivos y una de sus características en común es que tienen pelo.
- Grafos Conceptuales: existen dos tipos de nodos en estas redes: nodos

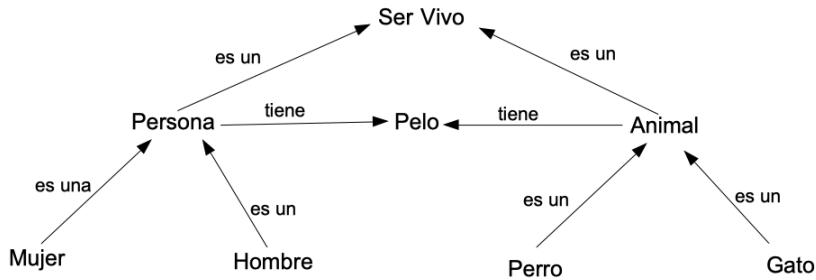


Figura 2.3: Ejemplo de Red de IS-A

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. Se puede ver un ejemplo en la Figura 2.4 en la cual la frase “*Man biting dog*” quedaría representada (Sowa, 1983). Los cuadrados implican el concepto y el círculo la relación entre ambos, por lo que en el caso de *man* y *bite*, la acción de morder la realiza *man* siendo éste el agente, y la relación entre *bite* y *dog* sería el objeto.



Figura 2.4: Ejemplo de Grafo Conceptual

Para el trabajo que queremos realizar, existen varias redes semánticas que disponen de una representación del conocimiento que necesitaremos para relacionar el concepto difícil introducido por el usuario con otros conceptos. A continuación, describiremos las más representativas.

2.1.1. ConceptNet

ConceptNet es una red semántica creada por el MIT (*Massachusetts Institute of Technology*) en 1999, fue diseñada para ayudar a los ordenadores a entender el significado de las palabras. ConceptNet ofrece la posibilidad de obtener de una palabra un listado de sinónimos, términos relacionados, términos derivados, el contexto de la palabra, resultados etimológicamente relacionados, símbolos, etc.

ConceptNet dispone de una aplicación web¹ donde se pueden buscar palabras

¹<http://conceptnet.io/>

en distintos idiomas, como el español, el inglés o el chino. En la Figura 2.5 se puede ver la información que devuelve la aplicación ConcepNet para la palabra chaqueta. Como se puede ver en la Figura hay una columna con los sinónimos del concepto en distintos idiomas (americana, *jacket* o *blouson*), otra columna con los términos relacionados (saco o chaquetear) también en distintos idiomas, otra columna que muestra los términos derivados del concepto (chaquetero, chaquetilla o chaquetita) y por último una columna que muestra otra forma de la palabra, en este caso el plural (chaquetas). Para algunos de los resultados de ConcepNet además aparece a la derecha de la palabra y entre paréntesis, una sola letra que indica si es un verbo, un sustantivo o un adjetivo.

The screenshot shows the search results for the Spanish term 'chaqueta' in ConceptNet 5.7. At the top, it says 'A Spanish term in ConceptNet 5.7'. Below that, there are links to 'Documentation', 'FAQ', 'Chat', and 'Blog'. The results are presented in four columns:

- Synonyms:** Includes terms like 'jacket (n, artifact)', 'jaqueta (n)', 'jacke (n)', 'americana (n)', 'casaca (n)', 'americana', 'casaca', 'jaqueta', 'chumpa (n)', 'jaqueta (n)', 'blouson (n)', 'veste (n)', and 'veston (n)'.
- Related terms:** Includes 'clothing', 'jacket', 'chaquetear', 'branlette', 'veste', 'veston', 'chaquear (v)', and 'saco'.
- Derived terms:** Includes 'chaquetero', 'chaquetilla', and 'chaquetita'.
- Word forms:** Shows the plural form 'chaquetas (n)'.

Figura 2.5: Resultados de ConcepNet para la palabra chaqueta

ConcepNet no realiza ninguna agrupación de los resultados en función del significado, si no que muestra todo el listado de palabras y en algunas de ellas aparece el contexto al cual se refieren. Por ejemplo, para la palabra arco algunos de los resultados que devuelve son referentes al significado arco de arquitectura y otros son referentes al arco de geometría.

Por otro lado, ConcepNet dispone de un servicio web² que devuelve los resultados en formato JSON. Siguiendo con la palabra chaqueta, se pueden ver en el Listado 2.1 los resultados devueltos por el servicio web para chaqueta. El JSON devuelto en este caso consta de cuatro campos principales³:

- @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.

²<http://api.conceptnet.io>

³<https://github.com/commonsense/conceptnet5/wiki/API>

- @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:
 - @id: describe el tipo de relación que existe entre la palabra introducida y la devuelta. En el Listado 2.1 se 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: puede ser la misma palabra buscada o una frase más completa, donde adquiera 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 porqué 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: algunos datos de ConceptNet se extraen de texto en lenguaje natural. El valor de surface text muestra lo que era este texto, puede que este campo tenga valor 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 los datos obtenidos.
- 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.

Listado 2.1: 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"
}
}
```

2.1.2. Thesaurus

Thesaurus⁴ es una aplicación web que se autodefine como el principal diccionario de sinónimos de la web. Esta página ofrece la posibilidad de obtener los sinónimos de una palabra, pero solamente devuelve resultados en inglés. Aparte del listado de sinónimos, Thesaurus indica que tipo de palabra es y una definición de la misma así como un listado de antónimos y un listado de palabras relacionadas con dicho concepto.

Si la palabra introducida es polisémica, Thesaurus devuelve los resultados por grupos según su connotación. En la Figura 2.6, se puede ver los resultados devueltos por Thesaurus para la palabra *book*, donde este puede utilizarse como un sustantivo o como un verbo. Y dentro de si es un tipo u otro, los divide según el significado, ya sea para utilizarlo como un documento, un diario o como una reserva. Y cada uno se muestran en pestañas distintas para facilitar al usuario la distinción de conceptos.

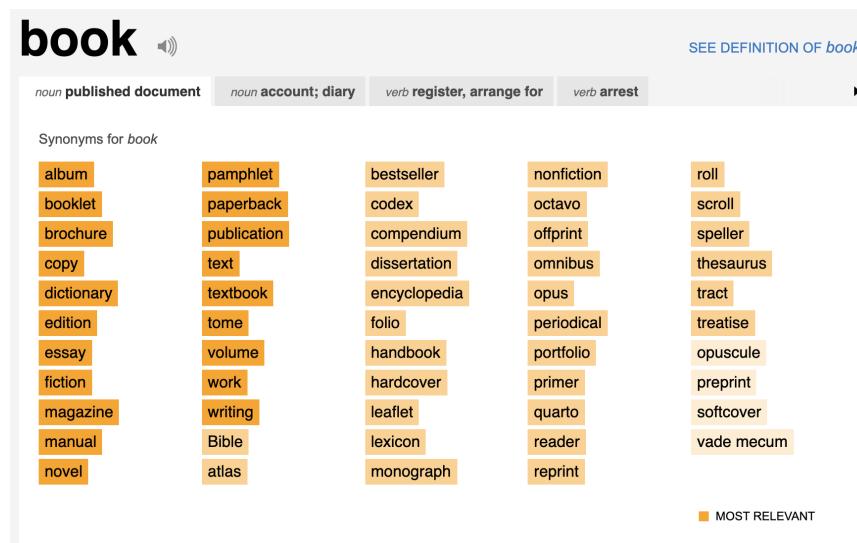


Figura 2.6: Resultados de ConcepNet para la palabra polisémica *book*

Por otro lado, esta aplicación proporciona una API⁵ tipo RESTful que obtiene los sinónimos de una palabra mediante una petición HTTP GET a la url <http://thesaurus.altervista.org/thesaurus/v1>. Este devuelve los resultados en formato XML o JSON. El contenido de la respuesta es una 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 |. Se puede ver en el Listado 2.2 el resultado devuelto por Thesaurus para la palabra *peace* en formato XML y en el Listado 2.3 para la misma palabra,

⁴<https://www.thesaurus.com/>

⁵<http://thesaurus.altervista.org/>

peace, pero en formato JSON. Ambos son muy similares, por ejemplo en formato XML se puede 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 aparecerá 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.

Listado 2.2: XML devuelto por Thesaurus para la palabra *peace*

```
<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> peace treaty | pacification | treaty | pact | accord </synonyms>
</list>
```

Listado 2.3: JSON devuelto por Thesaurus para la palabra *peace*

```
{
  "response": [
    {
      "list": [
        {
          "category": "(noun)", "synonyms": "order|war (antonym)"
        }
      ],
      "list": [
        {
          "category": "(noun)", "synonyms": "harmony|concord|concordance"
        }
      ],
      "lista": [
        {
          "categoria": "(noun)", "synonyms": "public security |security"
        }
      ],
      "lista": [
        {
          "categoria": "(noun)", "synonyms": "peace treaty | pacification | treaty | pact | accord"
        }
      ]
    }
  ]
}
```

2.1.3. Thesaurus Rex

Thesaurus Rex⁶ es una red semántica que solo admite palabras en inglés y que permite obtener las palabras relacionadas con una palabra o las categorías que comparten dos palabras.

En Thesaurus Rex las palabras tienen categorías y modificadores. Las categorías son sustantivos que definen a la palabra introducida por el usuario,

⁶<http://ngrams.ucd.ie/therex3/>

como se puede ver en la Figura 2.7, la palabra *house* tiene asociadas las categorías *structure*, *object*, *item* o *building*. Los modificadores son adjetivos que describen a la palabra, siguiendo con el mismo ejemplo, *house* tiene como modificadores *permanent*, *fixed* o *wooden*. Por último, dispone de un listado de las categorías más utilizadas por los hablantes de dicha lengua, como pueden ser *permanent-structure*, *inanimate-object*, *everyday-object*, etc...

En el caso de introducir dos palabras en la aplicación, por ejemplo *coffee* y *cola*, la aplicación devuelve las categorías que comparten dichos conceptos, en este caso serían *cold-beverage*, *dark-beverage* o *stimulating-beverage*. Si se introduce una palabra polisémica, como puede ser *book*, Thesaurus Rex no realiza ninguna distinción entre que resultados corresponden a los distintos significados del concepto buscado.

The screenshot shows the Thesaurus Rex interface. At the top, there is a logo of a green dragon-like creature with the text "THESSALONIUS REX". Below the logo is a search bar with the word "house" typed in. To the right of the search bar is a "Search!" button. Underneath the search bar, there is a message: "Use the & operator to see the shared categories of two terms. E.g. cola & coffee or divorce & war". Below this message are two buttons: "Go Back" and "See XML".

The main content area displays the search results for "house". It starts with a section titled "Top-Ranked Fine-Grained Categories of house:>Set as target =>see poetic categories". This section lists several categories:

- inanimate:object, private:residence, simple:object, common:object,
- permanent:structure, familiar:object, physical:structure,
- individual:object, permanent:installation, stationary:object,
- expensive:item, large:object, wooden:structure,
- permanent:construction, sensitive:area, tangible:thing, fixed:object, private:dwelling,
- occupied:structure, everyday:object, occupied:space, public:building, fixed:structure,
- static:object, real:object, architectural:building,
- physical:space, impervious:surface, large:item,

Below this, there are two tables:

Category Nuances for house:	Simple Categories for house:
simple, permanent, japanese, sensitive, fixed, surrounding, physical, wooden, old, lightweight, everyday, carpeted, public, main, stationary, sacred, urbanized, stationary, expensive, external, familiar, intellectual, sized, private, significant, world, base, exterior, luxurious, artificial, illegal, stable, immovable, flammable, visual, tall, size, informal, barren, historical, bulky, paved, open, minimalist, conspicuous, abandoned, discrete, framed, exclusive, relevant, concrete, off, oriented, convenient, historic, up, single, solid, anthropogenic, perceptual, quiet, boxed, municipal, straight, intrusive, unique, political, awkward, nonresidential	structure, housing, object, ground, activity, thing, item, ceremony, works, interior, decoration, community, point, toy, line, event, origin, building, space, residence, location, communication, fixture, pattern, part, creation, surface, unit, area, game, enterprise, establishment, set, grouping, theatre, class, behavior, personnel, action, county, disturbance, commodity, zone, measure, use, whole, job, facility, centre, organization, support, society, force, dwelling, cinema, article, entity, system, obstruction, installation, construction, business, land, marker, region, atmosphere, gathering, block, protection, agency, work, group, arrangement, artifact, monument, home, artefact, architecture, institution, cause, shelter, way, sector, sphere, body, stuff, formation, outside, attribute, side,

Figura 2.7: Resultados búsqueda Thesaurus Rex con la palabra *house*

La aplicación también dispone de una API pública que devuelve los resultados en formato XML. En el Listado 2.4 se muestra el XML devuelto

por la aplicación para la palabra *house*. El XML devuelto consta de tres grandes bloques: *Categories*, *Modifiers* y *CategoryHeads*. Todos los resultados tienen un peso (*weight*) asignado, esto significa que cuanto mayor sea el peso mayor es la similitud con el concepto dado. Los campos que se encuentran dentro del apartado *categories* son los resultados más utilizados en ese momento por los hablantes y que Thesaurus Rex ha encontrado, como por ejemplo *permanent-structure*. Los que se encuentran dentro de *modifiers*, como se ha comentado anteriormente son atributos del concepto a buscar, como por ejemplo *fixed*. Y por último, los que se encuentran en *categoryHeads* son las categorías más simples que se han encontrado para dicho concepto, como por ejemplo *structure*.

Thesaurus Rex utiliza el contenido de la web para generar sus resultados, con lo cual la información disponible no es fija, sino que varía según los datos 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 de manera semiautomática desde contenido de la web (Veale y Li, 2013). Por ejemplo, para la palabra *house*, uno de los resultados es *sphere*, que en principio no tiene ninguna relación con *house*.

Listado 2.4: XML devuelto por Thesaurus Rex para la palabra *house*

```
<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>
```

2.1.4. Metaphor Magnet

Metaphor Magnet es una aplicación web⁷, que crea metáforas a partir de una palabra y que solo está disponible para el inglés. Metaphor Magnet permite al usuario introducir palabras y, ayudándose de los n-gramas de Google (Veale y Li, 2012), busca e interpreta las distintas metáforas que existan sobre dicha palabra.

Un n-grama (Manning y Schütze, 1999) es una subsecuencia de n elementos consecutivos en una secuencia dada y a su vez estos pueden ser bigramas, trigramas, etc... Metaphor Magnet lo que hace es utilizando dichos n-gramas busca en la Web cuantas veces aparece el concepto introducido y cuantas veces aparece junto con otra palabra en un n-grama. De esta forma, se obtienen los resultados más utilizados por los hablantea. En la Figura 2.8 se puede ver el resultado obtenido en la aplicación para la palabra *house*. En este caso la aplicación devuelve métaforas propias del concepto, como *protecting:home*. En el caso de introducir una palabra polisémica, por ejemplo *book*, Metaphor Magnet no realiza ninguna agrupación según los distintos significados, si no que muestra todos los resultados juntos.

Metaphor Magnet también dispone de una API pública que devuelve los resultados en formato XML. En el Listado 2.5 se puede ver el resultado devuelto por la API para la palabra *house*. En el XML devuelto aparece la etiqueta *<Source Name>* seguido de la palabra a buscar, en este caso *house*. Otra etiqueta *<Score>* que muestra un número, cuanto mayor sea este número más relacionado será con el concepto introducido, por ejemplo "*tall:building*" tiene un *score* de 86 mientras que "*intuitive:natural*" tiene un *score* de 25, indicando así que es uno de los resultados menos relacionados con la palabra *house*. Por último, la etiqueta *<Text>* contiene la metáfora, por ejemplo "*protecting:home*".

Listado 2.5: XML devuelto por Metaphor Magnet para la palabra *house*

```
<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">
```

⁷<http://ngrams.ucd.ie/metaphor-magnet-acl/>

```

<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> powerful:locomotive </Text>
<Score> 57 </Score>
</Source>
<Source Name="house">
<Text> inspiring:manifesto </Text>
<Score> 41 </Score>
</Source>
<Source Name="house">
<Text> intuitive:natural </Text>
<Score> 25 </Score>
</Source>
</Metaphor>

```



New to Metaphor Magnet? Please see the README documentation.

[View XML](#)

[Go Back](#)

Source Metaphors: house

killing:abattoir, pious:clergymen, **towering:mountain**, packed:olive **protecting:home**,
 leading:general, **charming:castle**, organized:computer_network, **tall:building**, sprawling:forest,
 charming:kitten, threatening:pathogen, **beautiful:tree**, inspiring:vision, loving:puppy, threatening:enemy,
 satisfying:settlement, **strong:elephant**, connected:cellphone, cramped:bathroom, threatening:competitor,
charming:mansion, threatening:curse, crushing:seabuck, tending:vessel, loving:dally,
 entertaining:soap_opera, accountable:government, **charming:movie_star**, looming:crisis, threatening:cloud,
shining:star, calm:restrain, **strong:rock**, pink:skin, dreary:failure, registered:vehicle, charming:playboy,
 entertaining:carnival, charming:swindler, charming:illusion, solid:concrete, powerful:sledgehammer, inspired:idea,
 entertaining:magazine, respected:veteran, inspiring:gun, threatening:catastrophe, screaming:horn, swinging:club,
 appealing:privilege, attacking:despair, **shining:mirror**, haunting:reminder, entertaining:movie, *singing:movie*, loving:mercy,
 damaging:criticism, charming:lord, integrated:combination, inspiring:founder, protecting:key, caring:woman, charming:king,
drear:y:cave, entertaining:force, screaming:pig, threatening:monster, crowded:market, threatening:giant,
 charming:soundclip, commanding:controller, quaint:bistro, respected:expert, creeping:fungus, sparkling:sunbeam,
 attacking:party, healing:cure, threatening:thug, *respected*, shining:candle, beautiful:sunrise, *extinct:right_spiral*,
 attacking:belligerent, **protecting:fence**, threatening:danger, haunting:bated, charming:resort, threatening:mob,
 powerful:supercomputer, crowded:mall, fruitful:crop, smiling:junior, dignified:initial, dreary:tunnel, **trading:dealer**,
 wrecked:schooner, respected:yogi, teeming:ant hill, threatening:panther, **organized:religion**, entertaining:nightclub,
powerful:freight_train, inspiring:hyde, sweet:teddy, charming:gentleman, haunting:fear, **strong:bull**,
shining:skyscraper, respected:religious, teacher, inspiring:symbol, charming:kraze, **crowded:hive**,
 surprising:coincidence, shining:dew, **threatening:weed**, teaching:educator, sweet:sweetheart, damaging:glitch,
 comforting:balin, appealing:banquet, majestic:cow, exotic:aquarium, supporting:bond, charming:picnic, loving:husband,
 planning:developer, admired:sportsman, threatening:thunder_storm, reigning:bishop, loving:post, comforting:pillow,
 integrated:mixture, unwanted:pestilence, **big:barn**, unwanted:waste, understanding:confidante, teaching:consultant,
 entertaining:melodrama, loving:hug, light:bee, supporting:pole, **strong:tank**, attacking:gato, **shining:light**,
 inspired:madman, raving:fanatic, charming:lullaby, sweet:birthday, quaint:boutique, attacking:Mongol, dumb:gourd,
 striking:absurdity, inspiring:principle, loving:matrarch, charming:jewel, **reinforced:door**, sparkling:liquid,
 celebrated:shrine, entertaining:holiday, tender:fillet, *mignon*, pure:newborn, pioneering:revolutionary,
 dreary:depression, entertaining:game, teaching:coach, sparkling:writer, threatening:epidemic, charming:plaza,
 terrifying:challenge, charming:hunt, exciting:whirl, exciting:drama, attacking:thief, inspiring:art, handsome:joek,
 prestigious:penthouse, towering:rampant, respected:priest, sweet:moschud, happy:squirrel, reigning:grandmaster,
 inspiring:idol, pointed:pen, charming:fairy_tale, inspiring:inspiration, charming:luxury_hotel, light:feather, respected:dignitary,
 ruling:court, glowing:thunderbird, **packed:box**, charming:heartbeat, respected:thoroughbred, shining:brand, *soft:countryside*,
 beautiful:bimbo, **strong:stone**, intriguing:labyrinth, charming:gazebo, **shining:monument**, threatening:gun,
 inspiring:conqueror, fascinating:museum, **strong:horse**, powerful:computer, respected:wizard, entertaining:toy,
 towering:dragon, respected:professor, gloomy:pit, *singing:hoarse*, *turns*, fighting:fundamentalist, fast:fox, threatening:pitbull,
 fighting:defender, charming:actress, damaging:infestation, comforting:fleece, **entertaining:story**, smiling:kid,
 distinguished:minister, **manufacturing:firm**, inspiring:manifesto, loving:tribute, charming:babe, *showing:curves_postcard*.

Figura 2.8: Resultados búsqueda Metaphor Magnet con la palabra *house*

2.1.5. WordNet

WordNet es un *corpus*⁸ perteneciente a NLTK (*Natural Language Toolkit*)⁹ que almacena sustantivos, verbos, adjetivos y adverbios ignorando preposiciones, determinantes y otras palabras funcionales. Además está disponible en varios idiomas como el español, el inglés o el francés. Los conceptos se agrupan en conjuntos de sinónimos cognitivos llamados *synsets*, es decir, se agrupan según su significado. Por ejemplo, la palabra *guardia* tiene el siguiente grupo de sinónimos cognitivos para un *synset*: “defensor, guardián, protector y tutor” y para otro *synset* el siguiente grupo de sinónimos cognitivos: “velador, sereno, guarda de seguridad y vigilante”. A su vez, cada *synset* contiene:

- Hiperónimos: Palabras cuyo significado está incluido en el de otras¹⁰. Por ejemplo, *mamífero* es hiperónimo de *gato* y de *perro* ya que los gatos y los perros pertenecen al conjunto de los mamíferos.
- Hipónimos: Palabras cuyo significado incluyen el de otra¹¹. Por ejemplo, *gato* es hipónimo de *mamífero* ya que está incluido dentro del conjunto de los mamíferos.
- Holónimos: Palabras que representan el todo respecto a una parte. Por ejemplo, *coche* es el holónimo de *rueda*, *volante* y *acelerador* ya que forman parte de un todo, que es el coche.
- Antónimos: Palabras cuyo significado es totalmente opuesto al concepto inicial. Por ejemplo, el antónimo de *alegre* sería *triste*.
- Definición de la palabra: Oraciones cortas que explican el significado de la palabra. Esta información no aparece siempre.

Existen varias aplicaciones web que implementan dicho *corpus*, una de las más completas es EuroWordNet, ya que está disponible en varios idiomas y permite extraer sinónimos, antónimos, hiperónimos, hipónimos y holónimos. Además de algunas oraciones de ejemplo, y algunas definiciones de los *synsets* obtenidos. En la Figura 2.9 aparece la información devuelta por EuroWordNet para la palabra *casa*. Los datos obtenidos son:

- *Offset del synset*: es un código que lo identifica inequívocamente y que finaliza con una letra, la cual describe la categoría gramatical de los sinónimos devueltos. Por ejemplo, un *synset* de la palabra *casa* tiene asociado el *offset spa-30-02913152-n*, donde la “n” hace referencia a *noun*, esto quiere decir que es un sustantivo.

⁸Colección de documentos de texto

⁹Conjunto de bibliotecas y programas para el Procesamiento del Lenguaje Natural

¹⁰<https://dle.rae.es/?id=KRW1qe2>

¹¹<https://dle.rae.es/?id=KU5UAn5>

The screenshot shows the EuroWordNet interface with the search term 'casa' entered. On the left, a tree diagram of semantic relations is visible, with 'near_synonym' expanded to show 'edificio', 'inmueble', 'construcción', and 'edificación'. On the right, a search interface includes dropdowns for 'Look up' (set to 'Spanish_3.0') and 'Gloss' (set to 'Spanish_3.0'). Below these are checkboxes for 'Score', 'Rels', and 'Full'. A link to 'Category (ILI 3.0) - WikiMCR' is present. The main results area displays two entries:

- spa-30-02913152-n** 298 **edificio_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*;
- spa-30-03544360-n** 68 **casa_2** una vivienda que sirve de residencia para una o más familias; *el tiene una casa en Cape Cod;*

Figura 2.9: Resultados de la búsqueda en EuroWordNet para la palabra *casa*

- Sinónimos: listado con todos los sinónimos del *synset*. Por ejemplo en un mismo *synset* se obtiene edificio, inmueble, construcción y edificación.
- Definición: sirve para entender el concepto. Por ejemplo, en un mismo *synset* se obtiene la siguiente definición: “Una estructura que tiene un techo, paredes y se encuentra más o menos permanente en un solo lugar”.
- Ejemplo: En el caso de que la definición no sea suficientemente clara para el usuario, el ejemplo ayuda a comprender mejor el significado del concepto. Por ejemplo, para un *synset* se obtiene el siguiente ejemplo: “*Había un edificio de tres pisos en la esquina*”.

Si se introduce una palabra polisémica, EuroWordNet devuelve tantos *synsets* como significados tenga la palabra. Por ejemplo, para la palabra *portero* EuroWordNet devolverá en un *synset*: “conserje, guardabarrera y ostiario” haciendo referencia a portero como profesión y en otro *synset*: “arquero, guardameta, guardavalla y golero” que son los resultados para portero como jugador.

2.2. Lectura Fácil

Se llama lectura fácil¹² 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.

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 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 fácil¹³. En la actualidad, podemos distinguir los documentos en lectura fácil gracias al logo de la Figura 2.10.



Figura 2.10: Logo Lectura Fácil

Los documentos escritos en Lectura Fácil (Nomura et al., 2010) son documentos de todo tipo que siguen las directrices internacionales de la IFLA¹⁴ y de Inclusion Europe¹⁵ en cuanto al contenido y la forma. Algunas pautas a seguir para escribir correctamente un texto en Lectura Fácil son (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.

¹²<https://www.discapnet.es/areas-tematicas/diseno-para-todos/accesibilidad-de-comunicacion/lectura-facil>

¹³<https://dilofacil.wordpress.com/2013/12/04/el-origen-de-la-lectura-facil/>

¹⁴International Federation of Library Associations and Institutions

¹⁵Una asociación de personas con discapacidad intelectual y sus familias en Europa

- 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.
- 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.

Se debe también hacer hincapié en la distinción entre palabras fáciles y complejas (García Muñoz, 2012), puesto que son de gran importancia para la lectura fácil. Las palabras complejas son aquellas que no se utilizan a menudo, como por ejemplo: melifluo o inefable. Las palabras complejas deben descartarse en la lectura fácil, y en su lugar se deben introducir palabras fáciles, que son aquellas que se utilizan asiduamente. La RAE (Real Academia Española) dispone de un corpus¹⁶ donde se encuentran varios documentos en los que se incluyen las mil palabras más usadas, las cinco mil palabras más usadas y las diez mil palabras más usadas por los hablantes de lengua española. En estos listados se incluyen verbos, sustantivos, preposiciones, pronombres, adjetivos, etc. Para este trabajo, es fundamental la distinción entre palabras fáciles y palabras complejas, por ello los documentos facilitados por la RAE de las 1.000, 5.000 y 10.000 palabras más usadas se tendrán como referencia de palabras fáciles y por tanto se utilizarán para poder obtener los resultados de la aplicación. Aunque de todas las palabras de estos documentos, se trabajará únicamente con verbos, sustantivos y adjetivos.

¹⁶<http://corpus.rae.es/lfrecuencias.html>

2.3. Figuras retóricas

Las figuras literarias (o retóricas) se podrían definir como 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 (Galiana y Casas, 1994). Según la RAE (Real Academia Española)¹⁷, “*la retórica es el arte de bien decir, de dar al lenguaje escrito o hablado eficacia bastante para deleitar, persuadir o conmover*”. Las tres principales figuras retóricas son la metáfora, el símil y la analogía. Estas figuras se basan en la comparación de dos conceptos (Galiana y Casas, 1994): el 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.

En este trabajo se van a utilizar los tres tipos de figuras retóricas mencionados anteriormente (Calleja, 2017):

- Metáfora: Utiliza el desplazamiento de características similares entre dos conceptos con fines estéticos o retóricos. Por ejemplo, cuando el tiempo de una persona es muypreciado se dice: “Mi tiempo es oro”.
- 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”.

En nuestro trabajo, las figuras retóricas van a permitir comparar el concepto complejo introducido por el usuario con otros conceptos más sencillos.

2.4. 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

¹⁷ <https://dle.rae.es/?id=WISC3uX>

en el cual estén programadas o ejecutadas en cualquier tipo de plataforma¹⁸. Según el W3C (*World Wide Web Consortium*)¹⁹, “*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 (Torres, 2017):

- Es 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. Debe tener algún mecanismo que permita encontrarlo. De esta forma tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente.

2.4.1. Tipos de Servicios Web

Los servicios web pueden definirse tanto a nivel conceptual como a nivel técnico. A nivel técnico se pueden diferenciar dos tipos de servicios web (Torres, 2017):

- Servicios web SOAP (Simple Object Access Protocol): SOAP es un protocolo basado en XML para el intercambio de información entre ordenadores. Normalmente utilizaremos SOAP para conectarnos a un servicio e invocar métodos remotos²⁰. Los mensajes SOAP tienen el formato representado en el Listado 2.6, donde podemos ver un ejemplo para reservar un vuelo y está formado por los siguientes campos:
 - <Envelope>: elemento raíz de cada mensaje SOAP. Contiene dos elementos:
 - <Header>: es un elemento opcional que se utiliza para indicar información acerca de los mensajes SOAP. En el ejemplo del Listado 2.6 dentro del campo Header estarían los campos de reservas y pasajeros.

¹⁸<http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html>

¹⁹<https://www.w3.org/>

²⁰<https://www.ibm.com/support/knowledgecenter/es>

- <Body>: es un elemento obligatorio que contiene información dirigida al destinatario del mensaje. En el ejemplo del Listado 2.6 se puede ver los campos asociados a un itinerario, teniendo este el lugar de partida, de llegada, la fecha de llegada y la preferencia de asiento.
- <Fault>: es un elemento opcional para notificar errores. En el Listado 2.6 podemos ver que no se encuentra presente, pero en caso de ser utilizado deberá aparecer dentro del elemento *Body* y no puede aparecer más de una vez.

Listado 2.6: Estructura de un mensaje SOAP

```
<?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-pqff9fe8j7d</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>
```

- Servicios Web RESTful: RESTful es un protocolo que suele integrar mejor con HTTP que los servicios basados 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.

2.4.2. Arquitectura Servicios Web

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

- 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 capas²¹:

- Descubrimiento del Servicio: responsable de centralizar los servicios web en un directorio común, de esta forma es más sencillo buscar y publicar.
- Descripción del Servicio: como ya hemos comentado con anterioridad, los servicios web se pueden definir a sí mismos, por lo que una vez que los localicemos el Service Description nos dará la información para saber qué operaciones soporta y cómo activarlo.
- Invocación del Servicio: invocar a un Servicio Web 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.
- Transporte: todos los mensajes han de ser transmitidos de alguna forma entre el servidor y el cliente. El protocolo elegido para ello es HTTP.

2.4.3. Ventajas de los Servicios Web

Las principales ventajas del uso de los servicios web son las siguientes (Vega 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.

²¹<https://diego.com.es/introduccion-a-los-web-services>

- 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.

2.4.4. Desventajas de los Servicios Web

El uso de servicios web también tiene algunas desventajas²²:

- 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.

²²<http://fabioalfarocc.blogspot.com/2012/08/ventajas-y-desventajas-del-soap.html>

Capítulo 3

Herramientas Utilizadas

En este capítulo se van a explicar las herramientas utilizadas para el desarrollo de este trabajo. En el apartado 3.1 se explicará Django que es el *framework* utilizado para el desarrollo de la aplicación y en el apartado 3.2 se explicará SpaCy, que es la herramienta que se ha utilizado para la clasificación semántica de las palabras fáciles. Las herramientas y recursos que se han usado para la gestión del proyecto se explicarán en un capítulo aparte.

3.1. Django

Para construir un servicio web se necesita una manera de gestionar los elementos propios de estos servicios, como pueden ser el procesamiento de formularios y el mapeo de urls, para satisfacer estas necesidades, se requieren *frameworks* web que son estructuras que contienen los componentes necesarios para el desarrollo de aplicaciones web¹. Django es un *framework* de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles y se basa en el patrón MVC². Fue desarrollado entre los años 2003 y 2005 por un grupo de programadores que se encargaban de crear y mantener sitios web de periódicos. Es gratuito y de código abierto y dispone de una gran documentación actualizada así como muchas opciones de soporte gratuito y de pago.

Algunas de las razones por las que se ha elegido este *framework* han sido las siguientes³:

- Seguridad: Implementa por defecto algunas medidas de seguridad para evitar SQL Injection(adición de consultas SQL malignas que puedan

¹<https://tutorial.djangogirls.org/es/django/>

²<https://docs.djangoproject.com/en/2.0/>

³<https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>

alterar la base de datos) o Clickjacking(Que el usuario haga click en un enlace oculto, para obtener información del mismo sin su permiso o incluso tomar el control de su ordenador) por JavaScript.

- Escalabilidad: Se puede pasar de una aplicación sencilla a otra más compleja rápidamente, ya que es muy fácil añadir nuevos módulos al *framework*.
- Fácil acceso a bases de datos: Mediante ORM(Object Relational Mapper), que es una biblioteca para el acceso de datos, generando clases a partir de las tablas de la base de datos para poder realizar las consultas. Django tiene su propio ORM, con el que se pueden hacer consultas de manera muy intuitiva.
- Además, es muy popular por lo que para resolver cualquier problema que surja, como se ha explicado anteriormente, hay mucha documentación disponible y muchos hilos en foros de programación en donde encontrar posibles soluciones.

3.2. SpaCy

Cuando se obtuvieron las palabras fáciles de la RAE solo se necesitaban adverbios, nombres, verbos y adjetivos, por lo que se requería un análisis semántico de cada una de ellas para obtener solo las palabras que pertenecían a uno de estos grupos. Para ello se utilizó SpaCy. SpaCy⁴ es una biblioteca de código abierto para el Procesamiento del Lenguaje Natural en Python. Soporta más de 34 idiomas, entre ellos el español.

El analizador semántico de SpaCy, según un estudio realizado en 2015 por la universidad Emory⁵, es el más rápido y según Medium Corporation tiene un índice de acierto mucho mayor que el analizador sintáctico de NLTK⁶. Para utilizar en analizador, hay que importar la biblioteca de idioma correspondiente (en este caso español: “es_core_news_sm”) y pasar como parámetro las palabras que se deseen clasificar. El resultado para cada palabra introducida en el analizador, serán una serie de etiquetas, de todas ellas, la utilizada es la etiqueta “pos”, que contiene el grupo semántico de la palabra analizada, como se puede ver en la Figura 3.1, se ha introducido la frase: “el coche es rojo” y se han obtenido los siguientes resultados:

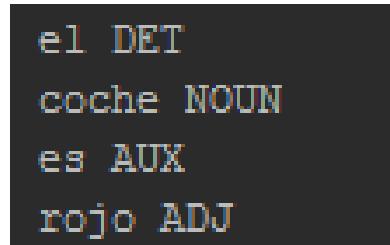
- el: ha obtenido la etiqueta DET, haciendo referencia a que es un determinante.
- coche: ha obtenido la etiqueta NOUN, que significa nombre en inglés.

⁴<https://spacy.io/>

⁵<https://www.aclweb.org/anthology/P15-1038>

⁶<https://medium.com/@pemagrg/private-nltk-vs-spacy-3926b3674ee4>

- es: ha obtenido la etiqueta AUX, haciendo referencia a que es un verbo auxiliar.
- rojo: ha obtenido la etiqueta ADJ, haciendo referencia a que es un adjetivo.



el DET
coche NOUN
es AUX
rojo ADJ

Figura 3.1: Ejemplo de clasificación de palabras

Se ha elegido ya que ha sido el que mejor resultado ha dado. Inicialmente se utilizó el POS-tagger (clasificador sintáctico de palabras) de NLTK, pero su índice de acierto no era muy bueno, por lo que se descartó su uso. A continuación se probó con SpaCy y ,además de ser más rápido, su índice de acierto ha sido prácticamente del 100 % ,por lo que fue lo finalmente utilizado.

Propuesta solución: Aplicación Aprende Fácil

4.1. Diseño centrado en el usuario de los prototipos de la Aplicación

Como ya se ha explicado en capítulos anteriores, esta aplicación está creada por y para personas que tienen alguna discapacidad cognitiva, por lo que el diseño de la interfaz debe ser sencillo y su uso no debe llevar a confusión ni debe hacer sentir al usuario frustrado al usarla. Es por ello que el diseño debe estar centrado en el usuario, y en nuestro caso aún más puesto que los usuarios finales requieren un diseño centrado en sus necesidades. Así pues, al ser un campo poco conocido por los integrantes del equipo, se ha necesitado de la ayuda de expertos para saber con mas detalle cual es el diseño adecuado. El diseño de la aplicación se ha realizado en dos iteraciones: una iteración competitiva entre los integrantes del grupo y una segunda iteración con expertos del Colegio Estudio3 Afanias¹.

4.1.1. Primera Iteración: Iteración Competitiva

En esta primera iteración cada integrante del grupo ha realizado su propio diseño de la aplicación. En la realización de estos diseños, los integrantes no podían hablar entre ellos ni comentar las diferentes ideas que tenían para la implementación. De esta forma se consigue que los diseños sean totalmente dispares y que las ideas de uno no provoquen la modificación del diseño del otro y que surjan ideas distintas. Se realizaron cuatro prototipos distintos ya que no teníamos claro si los usuarios finales iban a preferir que se mostrara un solo significado o todos, si mostrar la definición y el ejemplo les iba a

¹<https://afanias.org/que-hacemos/educacion/colegio-estudio-3/>

ayudar o no y si añadir los pictogramas les ayudaría o no. Los prototipos fueron los siguientes:

- Prototipo 1: Se muestra un único significado, siendo este el más común e incluyendo las comparaciones.
- Prototipo 2: Se muestra el significado más común, junto con una definición tradicional del concepto e incluyendo las comparaciones.
- Prototipo 3: Se muestran todos los significados de la palabra buscada e incluyendo las comparaciones.
- Prototipo 4: Se muestran todos los significados de la palabra y se añaden pictogramas² a las palabras, haciendo así que la comprensión del concepto sea mucho mas sencilla. Tanto en el prototipo diseñado por Pablo como en el de Irene se han utilizado los pictogramas de ARASAAC³.

En las Figuras 4.1, 4.2, 4.3 y 4.4 se muestran los prototipos creados por Pablo y en las Figuras 4.5, 4.6, 4.7 y 4.8 los prototipos creados por Irene.

Una vez que los prototipos estaban terminados, los juntamos para hacer una puesta en común y analizar los prototipos creados. Por lo general, los prototipos de los dos integrantes del grupo eran bastante similares, las principales diferencias eran:

- Ambos prototipos integran los mismos elementos: un campo de texto para introducir la palabra, un botón de búsqueda y los resultados se muestran en forma de lista, agrupando los distintos resultados en rectángulos, que a partir de ahora denominaremos fichas. Pablo ha optado por formas rectangulares, tanto para las fichas como para el campo de texto, mientras que Irene utiliza formas redondeadas en todos los elementos. Por otro lado, Pablo implementó un diseño orientado a niños por lo que para el color utilizó azules muy suaves y fondos juveniles con lápices de colores, gomas de borrar, etc. Irene utilizó un color mostaza, siendo un diseño más simple pero intentando abarcar a un usuario de cualquier edad. Por último, los resultados del prototipo de Irene, se muestran en color mostaza indicando de esta forma que se pueden pinchar en ellos, y se realizará la búsqueda del concepto pulsado.
- El orden a la hora de mostrar los resultados es distinto. En los prototipos de Pablo se muestran primero las metáforas (*Un vehículo es una máquina* o *Un vehículo es un transporte*), después los símiles (*Un vehículo es*

²Se entiende como pictograma un dibujo, imagen o figura que representa el significado de una palabra.

³<http://www.arasaac.org/>

como un taxi) y finalmente las analogías (*Un vehículo es rápido como un caballo*). En los prototipos de Irene se muestra únicamente una analogía, una metáfora y un símil para cada significado de la palabra buscada.

- En el prototipo 3 de Pablo incluye justo debajo de los resultados mostrados un ejemplo siempre visible (“Él necesita un coche para ir a trabajar”) para facilitar al usuario la comprensión del término buscado.
- En el prototipo 3 de Irene incluye justo debajo de los resultados mostrados una definición dentro de un botón (“Vehículo motorizado de cuatro ruedas por lo general impulsado por un motor de combustión interna”) dando a elegir al usuario la decisión de poder verla o no.
- Pablo decidió añadir el título “Un X es ...” antes de la lista de los resultados, englobando de esta forma los conceptos que tienen el mismo significado. Si existen varias acepciones del concepto, añade el título “O también puede ser...” en los siguientes, dejando claramente divididos los distintos significados que pudieran existir para el concepto buscado. En cambio Irene, añade un único título al principio (“Resultados para la palabra X”) donde queda claro que los resultados que se obtienen son para dicho concepto y además añade delante de cada metáfora: “Un X es...” y delante de cada símil y analogía: “Un X es como...”.
- Para la numeración de los resultados obtenidos, Pablo incluye dentro de cada ficha un listado numérico e Irene únicamente añade un número a la ficha.
- En el prototipo donde se incluyen también los pictogramas, Pablo añade un pictograma que hace referencia al concepto buscado según el contexto en que se utilice. Por ejemplo, un vehículo puede hacer referencia a un coche o puede ser un medio para llegar o lograr un fin, por lo que el añade el pictograma en función de su significado, e Irene además de incluir el mismo pictograma que Pablo, añade pictogramas a las palabras usadas en las comparaciones. Por ejemplo, en la frase “Un portero es un vigilante” Irene añadió el pictograma de vigilante.

Una vez analizados los prototipos de ambos integrantes nos reunimos con los directores del trabajo y se decidió crear un prototipo con las siguientes características:

- Para el diseño de la interfaz se eligió el de Irene por tener una interfaz más atractiva.
- Se eligió la palabra Portero para utilizarla como ejemplo en todos los prototipos.

- El orden de las resultados se modificó, haciendo que primero aparezcan las metáforas (“Un portero es un vigilante”), después los símiles (“Un portero es como un guardia”) y por último las analogías (“Un portero es tan ágil como un pájaro”).
- Se eliminó la palabra *tan* en la plantilla usada para las analogías.
- Se creó un nombre para la aplicación que estuviese en español: “Aprende Fácil”.
- En vez de mostrar un único símil, una metáfora y una analogía. Se deben mostrar todos los resultados obtenidos para dicho concepto.
- Se decidió que si la palabra solo tenía un significado se eliminaba el número de la ficha.
- Se añadió un pictograma a la característica que relaciona el concepto con el resultado en las analogías. Por ejemplo, en la frase *Un portero es tan fuerte como un gorila* se añadió el pictograma correspondiente a fuerte.
- Dejar el ejemplo del prototipo de Pablo junto con la definición del prototipo de Irene.
- Crear un nuevo prototipo donde la definición y el ejemplo se muestren a primera vista, y el usuario no tenga que estar pinchando en ningún botón para que los expertos nos indiquen cual es la mejor opción.
- Crear otro prototipo donde la definición y el ejemplo estén separados. Así el usuario puede elegir lo que quiere ver, si solo una cosa o ambas.

Con todas estas decisiones, se crearon los prototipos que se muestran en las Figuras 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 y que fueron los que se usaron para en la siguiente iteración donde nos reunimos con los expertos para que nos dieran su opinión sobre la aplicación, y nos ayudarán a decidir sobre los siguientes aspectos:

- ¿Se debe mostrar el significado más común o todos los significados de la palabra buscada?
- ¿Se debe añadir la definición tradicional y el ejemplo junto con las figuras retóricas o no? Y en caso de añadirlos, ¿Se deben mostrar juntos en un mismo botón, en distintos botones o sin botón y que aparezcan debajo de los resultados?
- ¿Son útiles los pictogramas? En caso de serlo, ¿solamente el pictograma de la palabra buscada o también añadimos los pictogramas de las palabras que se usan en las figuras retóricas?
- ¿La forma de mostrar los resultados es correcta, o es mejor otra disposición?

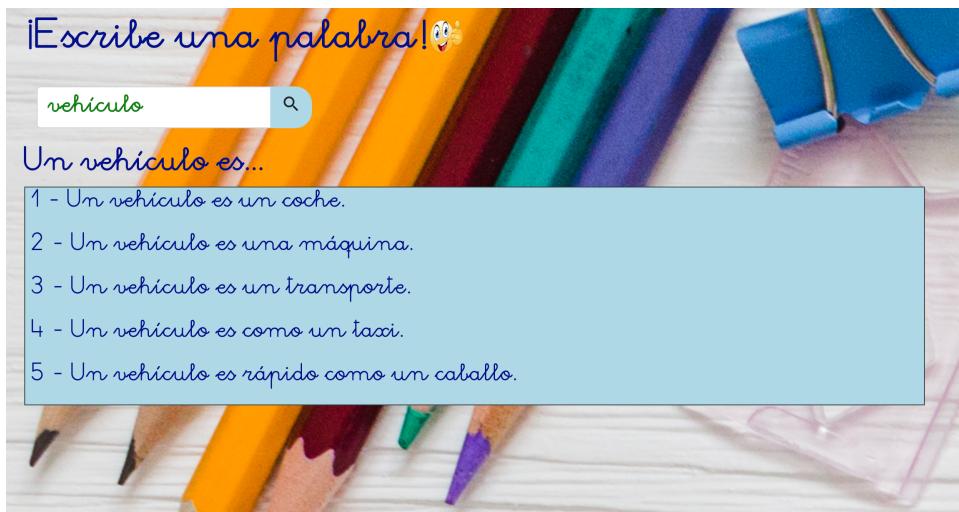


Figura 4.1: Prototipo de Pablo mostrando resultado más común para la palabra vehículo

4.1.2. Segunda Iteración: Evaluación con Expertos

El día 26 de Marzo de 2019 a las 09:00h nos reunimos con la directora y los profesores del colegio Estudio3 Afanias⁴ situado en la Comunidad de Madrid. Este colegio atiende a niños y jóvenes con discapacidad intelectual entre los 3 y 21 años de edad.

Lo primero que hicimos fue presentarles la aplicación y mostrarles los prototipos creados al final de la iteración anterior (Figuras 4.9, 4.10, 4.11, 4.12, 4.13, 4.14). Una vez expuestos los distintos prototipos, nos dieron la enhorabuena y nos hicieron participes de la gran ayuda que supondría esta herramienta. A continuación nos dieron su opinión sobre distintos aspectos que se podrían mejorar:

- Modificar el color amarillo-mostaza de los textos por un color más oscuro que contraste más con el fondo blanco. El color actual es morado oscuro para la barra que contiene el título de la aplicación y un violeta para resaltar los resultados obtenidos, así como para el botón de Definición y Ejemplo.
- El tipo de letra debe ser Arial o Script, ya que son las letras con las que los alumnos están familiarizados y las que mejor entienden.
- Añadir un reproductor que lea la frase, para hacer la aplicación accesible a aquellas personas que no puedan leer.

⁴<https://afanias.org/que-hacemos/educacion/colegio-estudio-3/>

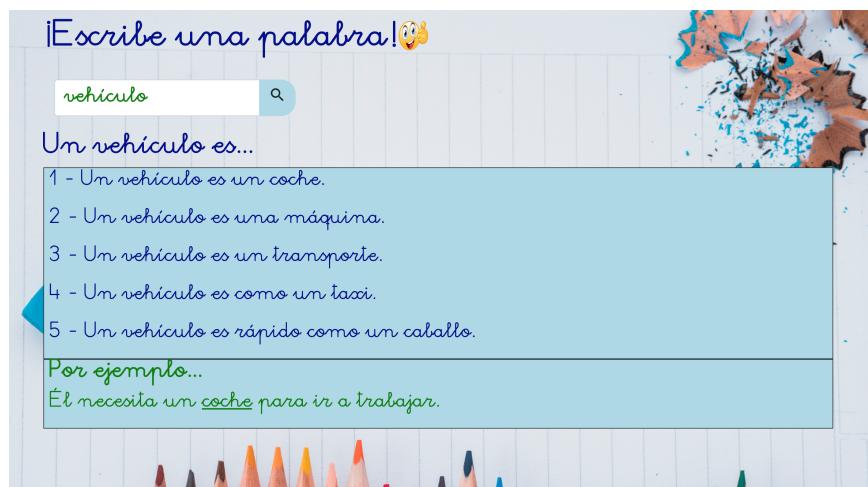


Figura 4.2: Prototipo de Pablo mostrando resultado más común para la palabra vehículo y un ejemplo

- Incluir la opción de poder ver un video para complementar la información devuelta por la aplicación.
- Los pictogramas deben situarse debajo de la palabra y no al lado, ya que los expertos comentaron que esto puede llevar a confusión a los alumnos pensando que sería otra palabra más para leer.
- Introducción de distintas personalizaciones:
 - Búsqueda en tres niveles: sencillo, medio y amplio. El nivel sencillo sería realizando la búsqueda de las palabras fáciles en las 1.000 palabras más usadas de la RAE, el medio en las 5.000 palabras más usadas de la RAE y el amplio en las 10.000 palabras más usadas de la RAE.
 - Añadir una opción que permita poner todos los textos en mayúsculas, haciendo la aplicación más accesible para aquellos alumnos que no entienden los textos en minúsculas.
 - Dejar que el usuario configure si quiere que aparezca la definición y el ejemplo o no.
 - Dejar que el usuario configure si aparecen los pictogramas o no.

Teniendo en cuenta las observaciones de los expertos se creó el diseño final de la aplicación.

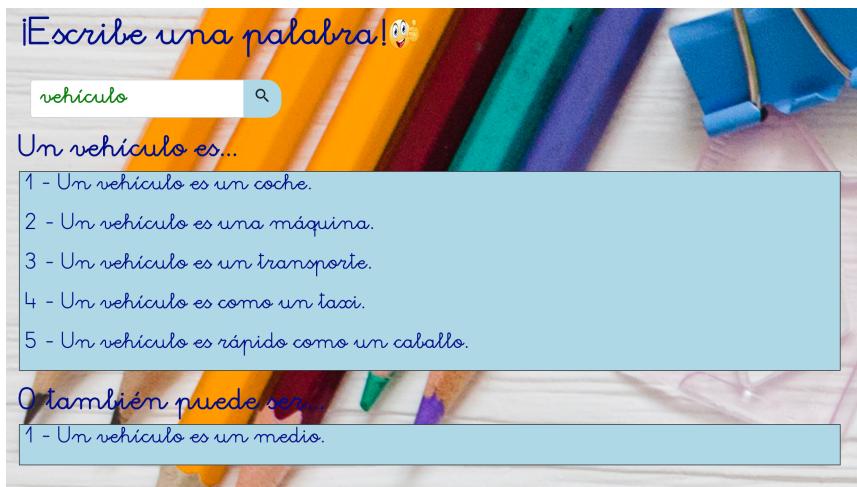


Figura 4.3: Prototipo de Pablo mostrando todos los resultados para la palabra vehículo

4.2. Arquitectura de Aprende Fácil

En este apartado se describirá la arquitectura diseñada para la aplicación. En la Figura 4.15 se puede ver los distintos componentes por los que está formada.

La aplicación está formada por una interfaz (*frontend*) la cuál realiza peticiones a la parte interna (*backend*), y este le devuelve la información correspondiente. Para ello, la parte *backend* debe realizar consultas a la base de datos. En la Figura 4.16 se puede ver la relación que existe entre el *frontend* con el *backend*, y los servicios web que son utilizados para la obtención de los resultados. Si el usuario introduce una palabra, dejando por defecto las distintas opciones que vienen y realiza la búsqueda de un concepto, la aplicación invocará a los servicios web relacionados con la obtención de metáforas (*getMetaphor*) y símiles (*getSimil*). A parte, si el usuario seleccionara la opción de “Mostrar definición y ejemplo”, el servicio web invocado sería el *getDefAndExample* y en caso de seleccionar la opción “Mostrar pictos” se invocaría al servicio web “*getImage*” para obtener el pictograma del resultado principal y “*getImagePalabra*” para obtener el pictograma de cada resultado mostrado.

Igualmente en los siguientes apartados se explicará con mayor detalle tanto la implementación del *frontend* como del *backend*.

Por último, la Universidad Complutense de Madrid nos facilitó un contenedor para poder alojar nuestro trabajo, accesible en la dirección holstein.fdi.ucm.es. Este servidor no tenía instalado ningún servicio por lo que se tuvo que realizar la instalación de todo lo necesario (python, pip, git, mysql, etc). Una vez



Figura 4.4: Prototipo de Pablo mostrando todos los resultados para la palabra vehículo junto con pictogramas

clonada la rama de GitHub donde tenemos el proyecto, el siguiente paso fue dejar el servidor siempre activo, para ello se tuvo que crear un servicio (*.service*) donde se guardaría la ruta del proyecto.

4.3. Diseño de la Base de datos de Aprende Fácil

Para este proyecto, se ha utilizado una base de datos para la persistencia de los datos. El sistema encargado de la gestión de la base de datos corresponde con MariaDB y esta constituida por las tablas que se pueden ver en la Figura 4.17 :

- Se ha hecho uso de la base de datos propia de WordNet, llamada MCR (*Multilingual Central Repository*)⁵, y se ha utilizado la versión 3.0. MCR es una base de datos de código abierto que integra distintas versiones de WordNet para seis lenguajes diferentes: Inglés, Español, Catalán, Vasco, Gallego y Portugués.

Se han utilizado principalmente cuatro tablas: “wei_spa-30_variant”, “wei_spa-30_relation”, “wei_spa-30_examples” y “wei_spa-30_synset”. De la tabla “wei_spa-30_variant” solamente se han utilizado estos atributos:

- *word*: Contiene la palabra. Cuando el usuario introduce la palabra en la aplicación, se realiza una búsqueda en dicha tabla para saber si la palabra buscada se encuentra .

⁵<http://adimen.si.ehu.es/web/MCR/>



Figura 4.5: Prototipo de Irene mostrando resultado más común para la palabra vehículo

- *offset*: Es el identificador de la palabra, aunque una misma palabra puede tener distintos *offsets* (uno por cada acepción).

De la tabla “wei_sp-30_relation”, solamente se han utilizado estos atributos:

- *sourceSynset*: Contiene el *offset* de hipónimo.
- *targetSynset*: Contiene el *offset* de hiperónimo.

De la tabla “wei_sp-30_examples”, se han utilizado los siguientes atributos:

- *offset*: Contiene el *offset* de la palabra.
- *examples*: Contiene el ejemplo correspondiente.

De la tabla “wei_sp-30_synset”, se han utilizado los siguientes atributos:

- *offset*: Contiene el *offset* de la palabra.
- *gloss*: Contiene la definición correspondiente.

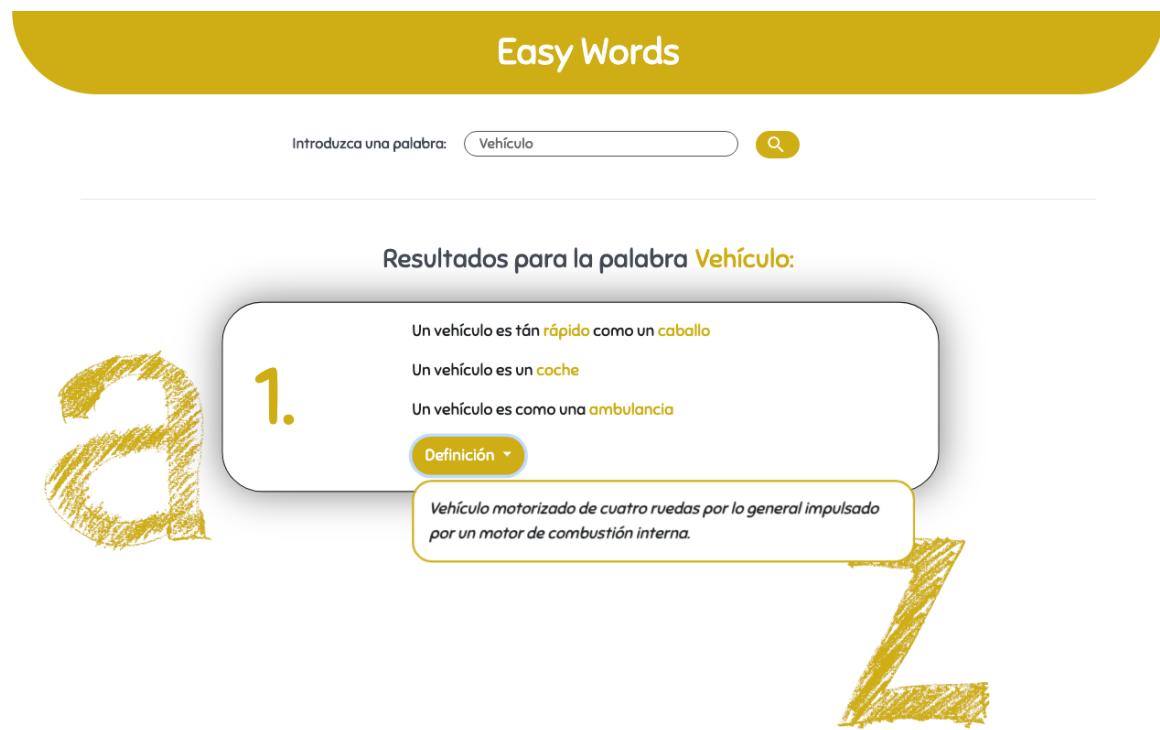


Figura 4.6: Prototipo de Irene mostrando resultado más común para la palabra vehículo y una definición

- En vez de tener tres ficheros donde guardar las 1.000, 5.000 y 10.000 palabras más usadas de la RAE, se guardaron los datos en tres tablas llamadas “1000_palabras_faciles”, “5000_palabras_faciles” y “10000_palabras_faciles”. Las tres se componen de dos columnas:
 - *word*: Contiene la palabra fácil.
 - *tag*: Contiene el tipo de palabra que es (sustantivo, adjetivo o verbo).
- Una tabla llamada “pictos”, que almacena la información de los pictogramas y está formada por las siguientes columnas:
 - *offset30*: Es el identificador del *synset* en la versión 3.0 de WordNet.
 - *palabra*: Contiene la palabra del pictograma.
 - *id_picto*: Es el identificador del pictograma.
 - *imagen*: Contiene el pictograma con una codificación binaria “base64”.



Figura 4.7: Prototipo de Irene mostrando todos los resultados para la palabra portero

4.4. Diseño del Backend de Aprende Fácil

4.4.1. Selección de la Red Semántica a utilizar

En este trabajo necesitamos una herramienta que nos proporcione las palabras relacionadas con un concepto dado y para ello vamos a emplear una red semántica. Como se ha comentado en el capítulo 2 existen varias redes semánticas que facilitan sinónimos, términos relacionados, metáforas, hiperónimos, etc. de un concepto dado. De todas las posibilidades presentadas en la sección 2.1 solo hay dos redes semánticas disponibles para el castellano: WordNet y ConcepNet. Para decidir con cual de las dos quedarnos hicimos una evaluación de ambas para evaluar si los conceptos devueltos por cada herramienta son correctos y sirven para nuestro propósito. En la sección 4.4.1.1 se explicará el diseño de la prueba que se ha realizado para ver que red semántica se utilizará finalmente en el proyecto, en la sección 5.2 se analizarán los resultados cuantitativos de la prueba realizada, en el punto 5.3, se analizarán los resultados obtenidos a nivel cualitativo y en el 5.4 se describirán las conclusiones finales de la evaluación. Por otra parte, en los

apartados posteriores, se detallarán los distintos servicios web utilizados para este proyecto, en la sección 5.5 se hablará de la base de datos utilizada y de su estructura, en los apartados 5.6, 5.7 y 5.8 se describirán los servicios web que se utilizan para la obtención de sinónimos, hipónimos e hiperónimos fáciles respectivamente. Por último, se hablará en los puntos 5.9 y 5.10 sobre los servicios web utilizados para crear las metáforas y los símiles que finalmente leerá el usuario.

4.4.1.1. Diseño de la evaluación

Para decidir que red semántica se iba a usar se decidió hacer una prueba con palabras que fuesen lo más heterogéneas posible, para ello se eligieron tres artículos periodísticos de distintos temas: tecnológico⁶, deportivo⁷ y político⁸. (Los artículos se pueden ver en el Apéndice A) De los artículos se seleccionaron únicamente los verbos, sustantivos y adjetivos, eliminando las palabras que estuvieran repetidas. Finalmente, quedaron 793 palabras para la realización de la prueba, para cada una de estas palabras se obtuvieron en WordNet y ConceptNet sus sinónimos y términos relacionados (en WordNet se entiende como término relacionado a los hiperónimos e hipónimos).

A continuación se comprobó de manera independiente cuantos de los sinónimos y términos relacionados se encontraban en cada una de las tres listas de palabras fáciles⁹ (análisis cuantitativo). Eligiendo como parámetros: porcentaje de palabras introducidas que no generan ningún resultado, porcentaje de palabras sin términos relacionados, porcentaje de palabras sin sinónimos y cantidad de palabras generadas por cada red semántica (WordNet y ConceptNet). Por último, se analizó la calidad de las palabras obtenidas por cada una de las redes semánticas en cada una de las listas de palabras fáciles, es decir, si las palabras que generaban tenían una relación aceptable con la palabra origen (análisis cualitativo). Eligiendo en ese caso como parámetros el porcentaje de sinónimos correctos y el porcentaje de términos relacionados correctos.

4.4.1.2. Resultados cuantitativos

En la Tabla 4.18 se muestran los resultados de la prueba cuantitativa realizada, como se puede observar, en términos generales, WordNet ofrece mejores resultados que ConceptNet ya que el porcentaje de palabras sin términos relacionados, sinónimos o ambas, es mayor en esta última. En el caso de la primera lista utilizada (1.000 palabras fáciles) ConceptNet no encontró ningún resultado para el 68,2 % de las palabras introducidas mientras que WordNet no encontró nada para el 62,6 %. En el caso de los

⁶https://elpais.com/elpais/2019/04/12/ciencia/1555061040_073105.html

⁷https://elpais.com/deportes/2019/04/22/actualidad/1555956020_022201.html

⁸https://elpais.com/politica/2019/04/23/actualidad/1556019953_831941.html

⁹listas de 1.000, 5.000 y 10.000 palabras más utilizadas en castellano según la RAE

términos relacionados, ConceptNet no encontró ninguno para el 68,75 % de las palabras, mientras que en el caso de WordNet fue del 24,72 %. Para el porcentaje de palabras sin sinónimos los resultados fueron de 88,06 para ConceptNet y 26,5 para WordNet. La siguiente lista probada fue la de 5.000 palabras fáciles, para la cual, como se ha explicado anteriormente, se han evaluado los mismos parámetros. Las palabras sin resultado han sido el 49,6 % para el caso de ConceptNet y 56,5 % en el caso de WordNet. El porcentaje de palabras sin términos relacionados ha sido de 47,15 % en ConceptNet y 16,19 % en WordNet. En los sinónimos, no se encontraron para el 77,84 % de las palabras utilizando ConceptNet y tan solo en un 15,05 % con WordNet. A continuación, utilizando la lista de 10.000 palabras fáciles, los resultados han sido: ConceptNet ha dejado un 43,4 % de palabras sin resultado frente al 55 % de WordNet, en los términos relacionados el 43,75 % usando ConceptNet frente al 15,05 % de WordNet. En los sinónimos, ConceptNet no ha encontrado resultados en el 76,7 % de las palabras y WordNet en el 9,2 %. Por último, se ha realizado un conteo de las palabras totales que ha generado cada red semántica, con un resultado de 2.652 sinónimos y términos relacionados generados por ConceptNet, mientras que WordNet ha generado 14.172.

4.4.1.3. Análisis cualitativo

Después del análisis cuantitativo, como se ha descrito en el apartado 4.4.1.1 se ha realizado el análisis cualitativo, que consiste en analizar una a una de manera manual, si las palabras obtenidas por ConceptNet y WordNet son correctas (entendiendo como correctas que tengan relación directa con la palabra buscada inicialmente). Como se puede observar, en la Tabla 4.19 los resultados son muy parejos para ambas redes semánticas. En el caso de la comparación con la lista de las 1.000 palabras fáciles, los sinónimos encontrados por WordNet son mejores que los de ConceptNet (65,45 % de sinónimos correctos frente a 46,15 %) sin embargo los términos relacionados son mejores en ConceptNet que en WordNet (78,67 % frente a 73,1 %). Después se hizo la comparación con la lista de 5.000 palabras fáciles, en la que ocurre lo mismo que con la lista anterior, hay más sinónimos correctos en WordNet (55,42 %) que en ConceptNet (55 %) y mejores términos relacionados en ConceptNet (84,17 %) que en WordNet (50,6 %). Por último en las comparaciones realizadas con las 10.000 palabras fáciles los resultados en ambos casos, fueron mejores en WordNet (60,73 % en el caso de los sinónimos y 55,3 % en los términos relacionados) que en ConceptNet (53,03 % y 53,19 % respectivamente).

4.4.1.4. Conclusiones

Valorando los datos obtenidos en ambas pruebas se puede concluir lo siguiente: de la prueba cuantitativa se puede deducir que la red semántica

más útil es WordNet ya que en 8 de los 10 parámetros valorados es mejor que ConceptNet. Ya que, aunque el porcentaje de palabras que dejan sin ningún resultado es muy parejo, el número de palabras que deja ConceptNet sin términos relacionados es mucho mayor que el de WordNet (68,75 % frente a 24,72 %, 47,15 % frente a 16,19 % y 43,75 frente a 15,05 %), ocurriendo lo mismo en el caso de los sinónimos (88,06 % frente a 26,5 %, 77,84 % frente a 15,34 % y 76,7 % frente a 9,2 %). En algunos de ellos ganando por una diferencia de más de 60 puntos, por ejemplo, en el porcentaje de palabras sin sinónimos utilizando la lista de las 10.000 palabras fáciles. Además, el número de palabras generadas por WordNet es casi 7 veces mayor que por ConceptNet (14.172 frente a 2.652) lo que supone un factor clave a la hora de elegir que red semántica utilizar finalmente ya que con una se pueden generar muchas más metáforas y símiles que con la otra.

Por otro lado, en la prueba cualitativa realizada, los resultados obtenidos son muy parecidos entre ambas redes semánticas, aunque una vez más, con la que se han obtenido mejores resultados ha sido WordNet ya que ha ganado 4 de los 6 parámetros medidos. Especialmente, en el caso de la lista de las 10.000 palabras, en donde gana ambas mediciones (53,03 % en el caso de los sinónimos correctos con ConceptNet frente a 60,73 % con WordNet y 55,19 % de términos relacionados correctos frente a 55,3 % con ConceptNet y WordNet respectivamente).

Por estos motivos, la red semántica que se ha decidido utilizar finalmente para la aplicación final ha sido WordNet.

4.4.2. Servicio Web para la obtención de Sinónimos Fáciles

Servicio web que recibe como entrada una palabra y un nivel de búsqueda, y devuelve todos los sinónimos en formato JSON.

La petición a este servicio sería GET y tendría la siguiente forma:

`https://holstein.fdi.ucm.es/tfg-analogias/easysynonym/json/word=palabra&level=nivel`

Donde *palabra* es el concepto a buscar y *nivel* es el grado de búsqueda, el cual puede tomar los siguientes valores:

- Nivel 1 (Nivel sencillo): Se filtran los sinónimos que están entre las 1.000 palabras más usadas de la RAE.
- Nivel 2 (Nivel medio): Se filtran los sinónimos que están entre las 5.000 palabras más usadas de la RAE.
- Nivel 3 (Nivel avanzado): Se filtran los sinónimos que están entre las 10.000 palabras más usadas de la RAE.

Una vez introducida la palabra y el nivel, se realiza una consulta a la base de datos a través de una *queryset* para obtener todos los *offsets* cuya palabra sea igual a la introducida. Si se obtienen resultados, se vuelve a buscar en

la base de datos para obtener las palabras de cada *offset*. Por cada palabra obtenida se comprueba que es distinta del concepto introducido y en función del nivel de búsqueda introducido, se realizará la búsqueda en una de las tres tablas que contienen las palabras fáciles de la RAE y comprobará si alguna de estas palabras se encuentra en dicha tabla.

Si la palabra se encuentra en las palabras más usadas de la RAE, se añade a su correspondiente campo del JSON, y si no se obtiene ningún resultado se devolverá el JSON solamente con un campo que guarda la palabra que se ha introducido.

Por ejemplo, si se realizará una búsqueda para la palabra “inmueble” con un nivel de búsqueda 2, la petición GET sería de la siguiente manera:

```
https://holstein.fdi.ucm.es/tfg-analogias/easysynonym/json/word=inmueble&level=2
```

Y en la Figura 4.21 se puede ver el resultado obtenido en formato JSON. En primer lugar aparece la palabra buscada, en este caso inmueble y a continuación un objeto que incluye tanto el *offset* como la lista de sinónimos fáciles .

4.4.3. Servicio Web para la obtención de hipónimos fáciles

La implementación de dicho servicio web se basa en que introduciendo una palabra y un nivel de búsqueda, este devuelve todos los hipónimos fáciles correspondientes en formato JSON. Para ello se realiza la siguiente petición GET:

```
http://127.0.0.1:8000/easyHyponym/json/word=palabra&level=nivel
```

Donde *palabra* es el concepto a buscar y *nivel* es el grado de búsqueda, igual que en el caso explicado anteriormente para buscar sinónimos fáciles.

Una vez introducida la palabra y el nivel, se realiza una consulta a la base de datos de MCR 3.0 a través de una *queryset* para obtener todos los *offsets* cuya palabra sea igual que la introducida. Por cada *offset* volveremos a buscar en la tabla *Relation* de la base de datos de MCR 3.0 para obtener los offsets que se encuentran en la columna *targetSynset*. Después, con cada *offset* obtenido de esta *queryset*, se realizará la búsqueda en la tabla *Variant* para obtener las palabras cuyo identificador sea igual que el offset de *targetSynset*. Mediante un cursor se realizará una búsqueda en una de las tres tablas de la RAE (en función del nivel introducido) y buscará si alguna de estas palabras se encuentra en dicha tabla. Si el resultado es positivo se añadirá al JSON.

Por ejemplo, si se realizará una búsqueda para la palabra inmueble con un nivel de búsqueda 2, la petición GET sería de la siguiente manera:

```
http://127.0.0.1:8000/easyHyponym/json/word=inmueble&level=2
```

Y en la Figura 4.22 se puede ver el resultado obtenido en formato JSON. En primer lugar aparece la palabra buscada, en este caso inmueble y a continuación un objeto que incluye el *offset* de hipónimo fácil , la lista de hipónimos fáciles así como el *offsetFather*, este identificador corresponde al

offset de uno de los *synsets* de inmueble. De esta forma, si una palabra buscada dispone de varios *synsets*, se podrá mostrar sus correspondientes sinónimos e hipónimos.

4.4.4. Servicio Web para la obtención de hiperónimos fáciles

La implementación de dicho servicio web se basa en que introduciendo una palabra y un nivel de búsqueda, este devuelve todos los hiperónimos fáciles correspondientes en formato JSON. Para ello se realiza la siguiente petición GET:

`http://127.0.0.1:8000/easyHyperonym/json/word=palabra&level=nivel`

Donde *palabra* es el concepto a buscar y *nivel* es el grado de búsqueda, igual que en los casos explicados anteriormente para buscar sinónimos e hipónimos fáciles.

Una vez introducida la palabra y el nivel, se realiza una consulta a la base de datos de MCR 3.0 a través de una *queryset* para obtener todos los *offsets* cuya palabra sea igual que la introducida. Por cada *offset* volveremos a buscar en la tabla *Relation* de la base de datos de MCR 3.0 para obtener los offsets que se encuentran esta vez y con diferencia de la búsqueda de hipónimos fáciles, en la columna *sourceSynset*. Después, con cada *offset* obtenido de esta *queryset*, se realizará la búsqueda en la tabla *Variant* para obtener las palabras cuyo identificador sea igual que el offset de *sourceSynset*. Mediante un cursor se realizará una búsqueda en una de las tres tablas de la RAE (en función del nivel introducido) y buscará si alguna de estas palabras se encuentra en dicha tabla. Si el resultado es positivo se añadirá al JSON.

Por ejemplo, si se realizará una búsqueda para la palabra inmueble con un nivel de búsqueda 2, la petición GET sería de la siguiente manera:

`http://127.0.0.1:8000/easyHyperonym/json/word=inmueble&level=2`

En la Figura 4.23 se puede ver el resultado obtenido en formato JSON. En primer lugar aparece la palabra buscada, en este caso inmueble y a continuación un objeto que incluye el *offset* de hiperónimo fácil, la lista de hiperónimos fáciles así como el *offsetFather*, ya explicado en el apartado anterior.

4.4.5. Servicio Web para la obtención de metáforas

Este servicio web dado un offset y una profundidad obtiene las metáforas de un concepto, es decir, se obtienen tanto los sinónimos como los hiperónimos fáciles formando la metáfora y este servicio devuelve dichos resultados en formato JSON. Para ello se realiza la siguiente petición GET:

`http://127.0.0.1:8000/metaphor/json/word=palabra&level=nivel`

Donde *palabra* es el concepto a buscar y *nivel* es el grado de búsqueda, igual que en los casos explicados anteriormente para buscar sinónimos, hipónimos

e hiperónimos fáciles.

La implementación de dicho servicio es idéntico al ya explicado en el servicio web para obtener sinónimos fáciles e hiperónimos fáciles, con la única diferencia que para poder formar la metáfora se llama a otro servicio implementado, el cual utiliza SpaCy para definir si el concepto es masculino o femenino y si está en singular o plural.

Por ejemplo, si se realizará una búsqueda para la palabra inmueble con un nivel de búsqueda 2, la petición GET sería de la siguiente manera:

```
http://127.0.0.1:8000/metaphor/json/word=inmueble&level=2
```

En la Figura 4.24 se puede ver el resultado obtenido en formato JSON. En primer lugar aparece la palabra buscada, en este caso inmueble y a continuación un objeto que incluye el *offset* del sinónimo o hiperónimo fácil, el *offsetFather* ya explicado en apartados anteriores y la lista de metáforas (es una casa, es una construcción, son unos edificios, es un edificio).

4.4.6. Servicio Web para la obtención de símiles

Este servicio web dado un offset y una profundidad obtiene los símiles de un concepto, es decir, se obtienen los hipónimos fáciles formando así el símil y devolviendo los resultados en formato JSON. Para ello se realiza la siguiente petición GET:

```
http://127.0.0.1:8000/simil/json/word=palabra&level=nivel
```

Donde *palabra* es el concepto a buscar y *nivel* es el grado de búsqueda, igual que en los casos explicados anteriormente. Para la implementación de dicho servicio, se utiliza exactamente el mismo código explicado en el apartado de obtención de hipónimos fáciles pero incluyendo la llamada al servicio de SpaCy para volver a definir si el concepto es masculino o femenino y si es singular o plural.

Por ejemplo, si se realizará una búsqueda para la palabra inmueble con un nivel de búsqueda 2, la petición GET sería de la siguiente manera:

```
http://127.0.0.1:8000/simil/json/word=inmueble&level=2
```

En la Figura 4.25 se puede ver el resultado obtenido en formato JSON. En primer lugar aparece la palabra buscada, en este caso inmueble y a continuación un objeto que incluye el *offset* del hipónimo fácil, el *offsetFather* ya explicado en apartados anteriores y la lista de símiles (es como una biblioteca es como un colegio, es como un restaurante).

4.5. Diseño del Frontend de Aprende Fácil

Igual que la parte interna de una aplicación es de suma importancia, la parte visual y como el usuario interactúa con ella es una parte a tener en cuenta siempre. Para este trabajo, como se ha comentado en la sección 4.1 se necesita aún más tener en cuenta cuales son los usuarios finales y sus

necesidades, por lo que la interfaz debe adecuarse a ellos. Lo primero que debemos decir respecto a la implementación, es que para la creación de los elementos visuales de la interfaz, así como para la colocación de los distintos elementos se ha uso de HTML, CSS y Bootstrap4 y para el funcionamiento de los distintos elementos así como para obtener los resultados que proporcionan los servicios web se ha hecho uso de JavaScript y jQuery.

En la Figura 4.26 se puede ver la interfaz de la aplicación final sin haber buscado aún ningún concepto. Se han numerado los elementos principales y a continuación se explicarán detalladamente:

- Elemento número 1 - Barra de navegación: Para la barra de navegación se ha utilizado la etiqueta `<nav>` propia de Bootstrap4 y para que los laterales queden redondeados se añadió al estilo la propiedad “radius” tanto en el borde inferior izquierdo como en el borde inferior derecho.
- Elemento número 2 - Contenedor: El contenedor engloba toda la página exceptuando la barra de navegación, para ello se añade la clase “container”, también propia de Bootstrap4 y dentro de este es donde se implementan el resto de elementos que forman la interfaz.
- Elemento número 3 - Selectores tipo “radio”: Para la implementación de este elemento, se creó un `<div>` para cada par de opciones de configuración (Por ejemplo, Convertir a minúsculas y Convertir a mayúsculas) añadiéndole una clase que contiene un estilo CSS para que los elementos se ordenen por columna, es decir, uno debajo de otro. Además se le añadió un “checkbox” de tipo radio para poder seleccionar una opción u otra pero no ambas a la vez.
- Elemento número 4 - Desplegable con opciones: Este desplegable se encuentra dentro de otra etiqueta `<div>` con una etiqueta `<select>` propia de Bootstrap4 y que permite añadir un listado con distintas opciones. En este caso sirve para que el usuario pueda decidir con qué nivel de búsqueda quiere realizar la consulta. A parte, se le añadió un estilo para que el elemento quede centrado en la página.
- Elemento número 5 - Entrada de texto y botón enviar: La barra para introducir la palabra es un `<input>` de tipo texto, al cual se le añadió una propiedad CSS “radius” para redondear los bordes y un “placeholder” con el texto “Palabra” para que el usuario sepa qué debe introducir. Y por otro lado, se encuentra el botón con un ícono de una lupa pero que al pasar el ratón por encima tiene un efecto y el botón se alarga, añadiendo a este ícono la palabra Buscar. Tanto el input como el botón están dentro de un `<div>` que contiene un estilo para centrar ambos elementos.

Se ha intentado separar mediante una barra el panel de arriba donde se encuentra la configuración de la aplicación y el panel abajo donde se muestran los resultados. Ahora que ya se ha explicado la implementación de los elementos estáticos de la interfaz, explicaremos como se crean de manera dinámica el resto de elementos. En la Figura 4.27 se encuentra la aplicación con los resultados para la palabra “Familia”. En esta Figura se han vuelto a enumerar los elementos para poder comprender mejor como funciona la aplicación. Para este ejemplo se ha seleccionado las opciones “Mostrar pictos” y “Mostrar definición y ejemplo”, y cuando se introduce una palabra y se da al botón de Buscar, se realiza una petición AJAX la cuál se conecta con el Backend y obtiene un JSON formado por cuatro objetos: todos los *offsets* de la palabra buscada, las metáforas, los símiles y las definiciones y ejemplos. Por cada *offset* se comprueba mediante JavaScript que *offset* de metáfora coincide y que *offset* de símil coincide y así poder mostrarlos en una misma ficha. A continuación, se explicarán como se construye cada elemento:

- Elemento número 1 - Texto informativo: Da información constante de la palabra que se ha buscado. En caso de que la palabra introducida sea inventada aparecerá “No hay resultados para la palabra X”. Este texto informativo se añade a la interfaz mediante jQuery una vez que se ha realizado la petición AJAX y se puede saber si tiene o no resultados.
- Elemento número 2 y 3 - Ficha: Como se ha explicado anteriormente, hay que mostrar los resultados según su significado, por lo que la relación entre el significado de la palabra buscada, y el significado de la palabra mostrada es el *offset*. Con JavaScript se recorre *offset* a *offset*, y por cada uno de ellos se comprueba que resultados tienen el mismo identificador. Esta ficha se crea dinámicamente según el número de resultados que se obtengan y es un <div>con una clase donde se añade un estilo para redondear los bordes, y la propiedad “box-shadow” para el sombreado, dando así sensación de que la ficha está superpuesta a la página. A parte, se debe saber si el resultado principal dispone de pictograma o no, por lo que se mediante la siguiente url “<https://holstein.fdi.ucm.es/tfg-analogias/imagen/offset>” donde *offset* es el identificador, se realiza una petición GET al servicio web correspondiente.
- Elemento número 4 - Resultado en formato lista: Como se puede observar los resultados se muestran en formato lista, primero aparecen las metáforas y después los símiles. Para poder obtener la frase completa, se recoge la metáfora (en este caso “familia es una grupo”) se realiza un *split* para poder quedarnos con la última palabra y se vuelve hacer una petición GET al servicio web con la siguiente url “<https://holstein.fdi.ucm.es/tfg-analogias/imagenByPalabra/palabra>” donde *palabra*, siguiendo con el

ejemplo anterior sería “grupo” y así se obtiene el pictograma correspondiente.

- Elemento número 5 - Botón definición y ejemplo: Se muestra la definición y ejemplo cuyo *offset* concida con el *offset* de la ficha principal. En caso de que no tenga se mostrará “NO HAY DEFINICIÓN NI EJEMPLO”. Este botón siempre aparecerá después de mostrar todas las metáforas y símiles.

Cabe destacar que una vez mostrados los resultados, tenemos la posibilidad de ocultar o mostrar los pictogramas así como la definición y el ejemplo sin tener que volver a realizar la búsqueda. Esto se consigue teniendo un manejador de eventos, el cual una vez que se pincha sobre los diferentes “checkbox” se añaden o eliminar las clases necesarias para que la interfaz se vea siempre correctamente. Por ejemplo, si en la Figura se pulsara sobre “Ocultar pictos”, estos desaparecerían y la palabra se quedaría perfectamente alineada con el texto de la metáfora. En el caso de pinchar sobre “Convertir a mayúsculas” se le añadiría un estilo CSS a la etiqueta <body>con la propiedad “text-transform: uppercase”. Por último, también se puede realizar la búsqueda de algún resultado obtenido pinchando sobre él, haciendo así que se vuelva a realizar una petición AJAX, recogiendo los resultados que se han comentado anteriormente y siguiendo el mismo proceso para mostrar los resultados.

4.6. Evaluación de Aprende Fácil

El día 17 de Mayo de 2019 tuvo lugar la evaluación final en el colegio Estudio3 Afanias situado en la Comunidad de Madrid. A esta evaluación acudieron los integrantes del equipo junto con los directores. Esta evaluación se dividió en dos grupos distintos: El primer grupo formado por 7 alumnos de la edad de 17 años y 1 profesor, y un segundo grupo formado por 7 alumnos de la edad de 14 años y 1 profesor. Cabe destacar que el primer grupo estaba formado mayoritariamente por alumnos autónomos e independientes, en contraposición del segundo grupo que no disponían de las mismas habilidades. Se utilizó un formulario de Google Forms dividido en dos secciones: Una primera sección guiada enfocada al análisis de los resultados, viendo si estos eran útiles y correctos. Y otra sección dedicada a la usabilidad de la aplicación. El formulario empleado en esta evaluación puede encontrarse en nuestro repositorio de GitHub, y en el Ápendice ..X..

La idea principal era que cada usuario probará la aplicación y realizara el formulario de forma individual, pero finalmente se decidió que los integrantes del equipo fueran los que realizaran la evaluación *in situ* mientras los alumnos daban su opinión y se recogían sus respuestas. A continuación explicaremos los resultados obtenidos en cada grupo:

Lo primero que se hizo fue introducir la palabra PUENTE con un nivel de búsqueda fácil, donde los alumnos comprendieron todos los resultados y les pareció bastante útil.

Después, se introdujo la misma palabra pero con un nivel de búsqueda medio. En este caso hubo que explicar el resultado “circuito” puesto que no sabían su significado en este contexto, y en el resultado “arco” no entendían que relación existía con el concepto introducido.

A continuación, se añadió la misma palabra pero con nivel de búsqueda avanzado. Para el resultado “placa”, en un primer momento no entendían el significado, pero una vez explicado lo comprendieron.

La siguiente palabra que se introdujo fue DICTADOR con un nivel de búsqueda medio, y de los resultados obtenidos comprendieron todos menos “soberano”. Y de los demás resultados obtenidos, no entendían la diferencia que existía entre “gobernador” y “gobernante”, lo que les llevó a confusión no dejando claro el significado del concepto.

Posteriormente, se procedió a realizar la prueba para obtener varios resultados. Por ello, se introdujo la palabra DEFENSA con un nivel de búsqueda fácil. Esta vez, se obtuvieron seis fichas con resultados, de las cuales entendieron todos. Para profundizar aún más, en esta búsqueda se añadió la opción de mostrar pictogramas, la respuesta de los alumnos fue que los resultados se entienden de una manera más clara y sencilla.

Después, se añadió la palabra DANZA con un nivel de búsqueda avanzado. En este caso, a todos los alumnos les quedó muy claro los resultados obtenidos.

La siguiente prueba, mostraba únicamente las metáforas y se introdujo la palabra PORTERO con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas. Los resultados obtenidos fueron correctos excepto para el resultado “funcionario” y “administrador”, que no entendían su significado. Después, se cambió el nivel de búsqueda a medio, para saber si con un nivel inferior donde se obtendrían menos resultados, les era más sencillo entenderlo. De esta forma se obtuvieron dos resultados, de los cuales el primero lo entendieron y el segundo solamente si se añade la opción de mostrar pictogramas.

A continuación, se realizó la prueba de obtener símiles, donde se buscó la palabra FUNCIONARIO con un nivel de búsqueda fácil. Los alumnos entendieron todos los resultados obtenidos.

Para probar la opción de mostrar definición y ejemplo, se buscó la palabra FAMILIA con un nivel de búsqueda fácil. De los resultados obtenidos, en la primera ficha no entienden el resultado “clase” pero si les ayuda la definición y ejemplo que aparece. Igual ocurre en la ficha tres pero con el resultado “línea”.

Con este mismo concepto, se realizó el cambio de minúsculas a mayúsculas, dejando claro que prefieren esta opción puesto que están más familiarizados y es como aprenden a escribir.

Por último, se llevó a cabo la sección de uso libre, donde los alumnos decían palabras para buscar y nos daban su opinión de los resultados:

- PASADO en nivel búsqueda fácil, los resultados obtenidos los entendieron perfectamente. - ENAMORARSE, no mostraba ningún resultado puesto que es un verbo conjugado y en la aplicación solo se pueden introducir infinitivos, por lo que se buscó ENAMORAR. Se obtuvo un resultado y lo entendieron porque se mostraba también el pictograma. - FIESTA en nivel de búsqueda avanzado. El resultado obtenido “suceso” les confunde un poco. Añadimos la opción de mostrar definición y ejemplo para intentar que les aclara pero les confundía más. En cambio, con la otra definición y ejemplo les aclaraba la idea del concepto. - DETECTIVE en nivel de búsqueda avanzado. Entienden todos los resultados excepto la palabra “paco”. De los resultados obtenidos, se pincho en la palabra INVESTIGADOR para realizar la búsqueda del concepto y entendían todos los resultados. - POLICÍA con nivel de búsqueda avanzado. No entienden el resultado “dotación” ni “tira”. - SOSO con nivel de búsqueda avanzado. El pictograma que se obtiene para el resultado “pesado” no tiene relación con el significado en dicho contexto. - SIRENA con nivel de búsqueda avanzado. Los resultados obtenidos los entendían perfectamente, aunque echaban en falta el resultado relacionado con ser mitológico.

Por último, les hicimos unas preguntas sobre la usabilidad. Las preguntas fueron las siguientes: - ¿La utilizarían habitualmente? Su respuesta fue afirmativa, argumentando que les sirve para poder explicar conceptos más complejos que no entienden. - ¿Prefieren que la opción de mostrar pictogramas sea configurable? SI - ¿Prefieren que la opción de mostrar definición y ejemplo sea configurable? SI - ¿Prefieren que la opción de convertir a mayúsculas y minúsculas sea configurable? SI - ¿Prefieren que los pictogramas sean a color o en blanco y negro? Prefieren a color - ¿Les gusta el color de la aplicación? SI - ¿Les resulta fácil de usar la aplicación? SI

En definitiva, nos dijeron que les gustó mucho la aplicación.

En el segundo grupo, se intentó seguir el mismo proceso utilizado con el primer grupo pero debido a que los alumnos eran más pequeños y se desconcentraban con mayor facilidad, solo pudimos seguir la parte guiada en unas cuantas preguntas y el resto se realizó con el profesor en ese momento presente.

Se introdujo la palabra PUENTE con nivel de búsqueda fácil. Les quedó claro el concepto pero añadiendo pictogramas. Realizando la misma búsqueda pero con nivel medio y añadiendo la opción de mostrar pictogramas, entendían los resultados obtenidos. Y con nivel avanzado, en el resultado “circuito” entienden su significado como “circuito de carreras” pero no el de electricidad, que era el obtenido.

A continuación, se añadió la palabra DICTADOR con nivel de búsqueda medio. Y entendían la palabra dictador como persona que dicta un texto a alguien. Y con nivel de búsqueda avanzado, no sabían qué es el resultado

“soberano”.

Para la palabra DEFENSA con nivel de búsqueda fácil, los resultados obtenidos son bastante complicados para los alumnos. En la primera ficha, les confundía la palabra “construcción”, aunque explicándoselo si sabían lo que era.

Con la palabra DANZA en nivel de búsqueda avanzado, entendían todos los resultados obtenidos. Después, se buscó la palabra PORTERO con nivel de búsqueda avanzado. Los alumnos saben los significados que están relacionados con portero de deporte y de profesión, pero la palabra “funcionario” y “administrador” no lo entienden. A parte, echaban en falta el resultado de portero automático.

A partir de este momento, la prueba se realizó con el profesor. Nos comentó que existían ciertas palabras que no saben como explicar su significado. Una de ellas fue TALAR, la cuál la buscamos en la aplicación con nivel de búsqueda avanzado. El resultado obtenido era correcto pero nos comentó que faltaba el pictograma que representa el concepto en sí. Y del resultado que se obtuvo (“talar es cortar”), el pictograma de cortar no era el adecuado según su acepción porque el mostrado era cortar en vez de talar.

Otra palabra que se buscó fue SACUDIR con nivel de búsqueda avanzado. Los resultados obtenidos algunos fueron correctos y otros no. De los correctos, el profesor indicó que debería aparecer primero el resultado más común o el mejor define el concepto (en este caso mover antes que batir), y otros resultados como trasladar y reducir no deberían aparecer por que no están relacionados con el concepto buscado. La definición y ejemplo que aparecen son correctos.

Nos comentó también que prefiere que la aplicación este en mayúsculas, pero que sea configurable. Y los pictogramas igual.

Easy Words

Introduzca una palabra: Portero 

Resultados para la palabra Portero:

1.  Un portero es tán fuerte como un gorila 

Un portero es un vigilante 

Un portero es como un guardia 

2.  Un portero es tán ágil como un pajaro 

Un portero es un jugador 

Un portero es como un futbolista 

Figura 4.8: Prototipo de Irene mostrando todos los resultados para la palabra portero junto con pictogramas

The screenshot shows a user interface for a learning application. At the top, a yellow header bar contains the text "Aprende Fácil". Below this, there is a search bar with the placeholder "Introduzca una palabra:" followed by a text input field containing the word "Portero". To the right of the input field is a magnifying glass icon. A horizontal line separates this from the main content area. In the content area, the heading "Resultados para la palabra Portero:" is displayed. To the left of this heading is a large, stylized letter "a" with a yellow pencil sketch texture. To the right is a large, stylized letter "Z" with the same texture. Within a rounded rectangular box, there are four lines of text: "Un portero es un **vigilante**", "Un portero es como un **guardia**", "Un portero es:
 fuerte como un gorila", and " **grande como un oso**".

Figura 4.9: Prototipo mostrando resultado más común para la palabra portero



Figura 4.10: Prototipo Versión 1 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo separados

Aprende Fácil

Introduzca una palabra: Portero

Resultados para la palabra Portero:

Un portero es un **vigilante**
Un portero es como un **guardia**
Un portero es:
 fuerte como un gorila
 grande como un oso

Definición: Un portero es alguien que protege una entrada.

Ejemplo: Mi tío trabaja de portero en una discoteca.

a

Z

Figura 4.11: Prototipo Versión 2 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo a simple vista

The image shows a user interface for a learning application. At the top, a yellow bar contains the title "Aprende Fácil". Below it, a search bar has the placeholder "Introduzca una palabra:" followed by the word "Portero" and a magnifying glass icon. A large yellow letter "a" is on the left, and a large yellow letter "Z" is on the right. A main content area displays the results for the word "Portero": "Resultados para la palabra Portero:". Inside a rounded rectangle, there are several definitions and examples. One example is highlighted with a blue button labeled "Definición y Ejemplo ▾". Below this button is a box containing the definition: "Definición: Un portero es alguien que protege una entrada." and the example: "Ejemplo: Mi tío trabaja de portero en una discoteca."

Aprende Fácil

Introduzca una palabra: Portero

Resultados para la palabra Portero:

Un portero es un **vigilante**
Un portero es como un **guardia**
Un portero es:
 fuerte como un gorila
 grande como un oso

Definición y Ejemplo ▾

Definición: *Un portero es alguien que protege una entrada.*
Ejemplo: *Mi tío trabaja de portero en una discoteca.*

a

Z

Figura 4.12: Prototipo Versión 3 mostrando resultado más común para la palabra portero junto con una definición y un ejemplo en un mismo botón

Aprende Fácil

Introduzca una palabra: Portero 

Resultados para la palabra Portero:

1.

Un portero es un **vigilante**
Un portero es como un **guardia**
Un portero es:
fuerte como un gorila
grande como un oso

2.

Un portero es un **jugador**
Un portero es como un **futbolista**
Un portero es:
rápido como un caballo
delgado como un flamenco

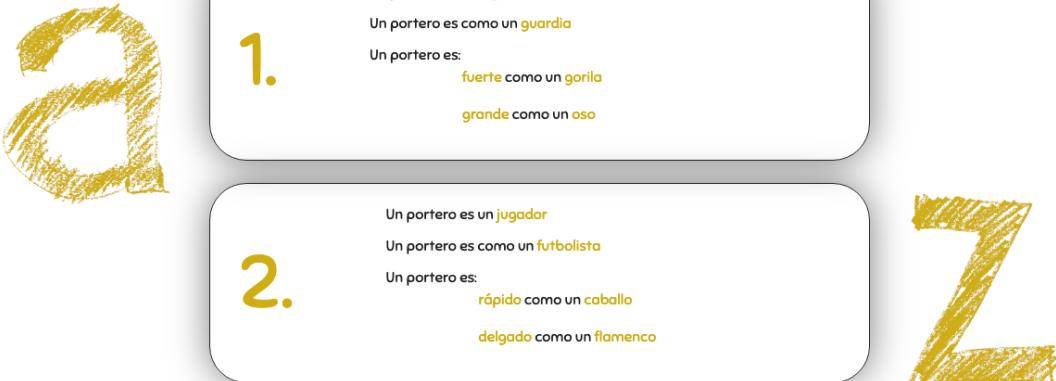


Figura 4.13: Prototipo mostrando todos los resultados para la palabra portero

Aprende Fácil

Introduzca una palabra: Portero

Resultados para la palabra Portero:

1.

Un portero es un **vigilante** 

Un portero es como un **guardia** 

Un portero es:
fuerte  **como un gorila** 

grande  **como un oso** 

2.

Un portero es un **jugador** 

Un portero es como un **futbolista** 

Un portero es:
rápido  **como un caballo** 

delgado  **como un flamenco** 

Figura 4.14: Prototipo mostrando todos los resultados para la palabra portero incluyendo los pictogramas

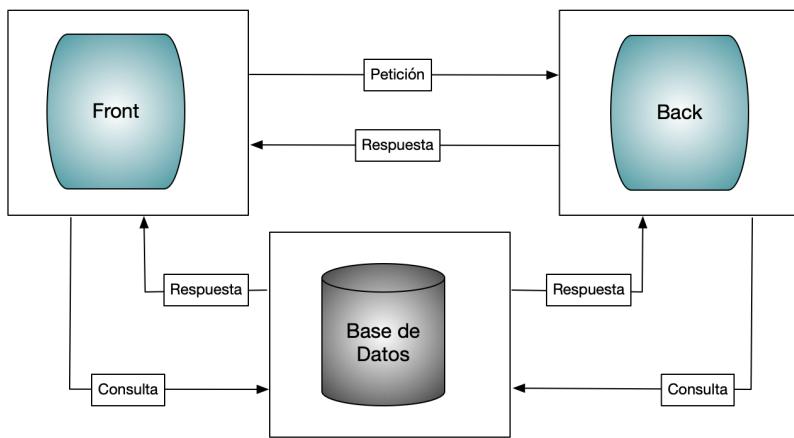
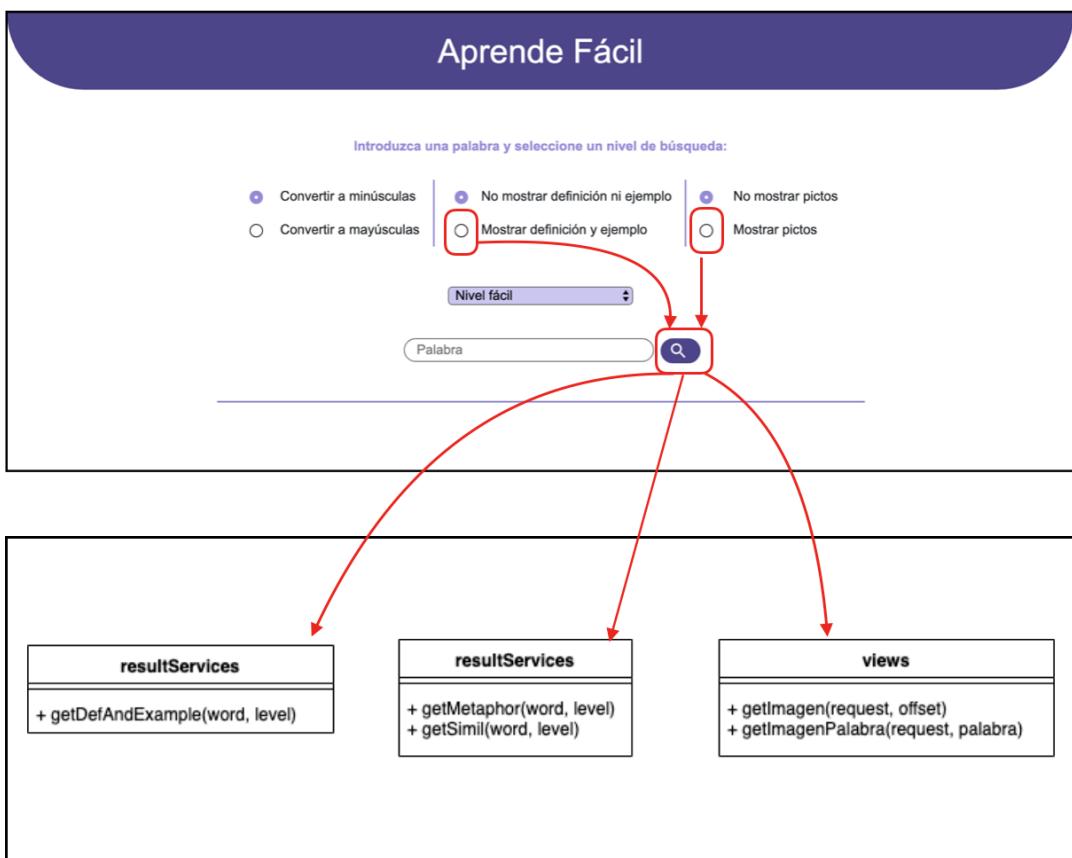


Figura 4.15: Diagrama de la arquitectura del proyecto

Figura 4.16: Relación entre *Backend* y *Frontend*

mcr30_wei_spa-30_variant

word : varchar(100)
sense : int(1)
offset : varchar(17)
pos : char(1)
cscs : float
experiment : varchar(20)
mark : varchar(20)

mcr30_pictos

offset30 : varchar(17)
palabra : varchar(100)
id_picto : int(10)
imagen : blob

mcr30_1000_palabras_faciles

word : varchar(100)
tag : varchar(100)

mcr30_5000_palabras_faciles

word : varchar(100)
tag : varchar(100)

mcr30_10000_palabras_faciles

word : varchar(100)
tag : varchar(100)

mcr30_wei_spa-30_relation

relation : smallint(6)
sourceSynset : char(17)
sourcePos : char(1)
targetSynset : char(17)
targetPos : char(1)
cscs : float
method : char(2)
version : char(1)
wnSource : char(4)

mcr30_wei_spa-30_examples

word : varchar(100)
sense : int(11)
examples : varchar(255)
pos : char(1)
offset : varchar(17)

mcr30_wei_spa-30_synset

offset : varchar(17)
pos : char(1)
sons : int(1)
status : enum('i','n')
lexical : enum('l','n')
instance : tinyint(1)
gloss : text
level : int(1)
levelFromTop : int(1)
mark : varchar(20)

Figura 4.17: Tablas utilizadas en la Base de Datos

		Conceptnet	Wordnet
Lista 1.000 palabras	% palabras sin resultado	68,2	62,6
	% palabras sin términos relacionados	68,75	24,72
	% palabras sin sinónimos	88,06	26,5
Lista 5.000 palabras	% palabras sin resultado	49,6	56,5
	% palabras sin términos relacionados	47,15	16,19
	% palabras sin sinónimos	77,84	15,34
Lista 10.000 palabras	% palabras sin resultado	43,4	55
	% palabras sin términos relacionados	43,75	15,05
	% palabras sin sinónimos	76,7	9,2
TOTAL PALABRAS GENERADAS		2.652	14.172

Figura 4.18: Tabla de resultados de la prueba cuantitativa

		ConceptNet	WordNet
Lista 1.000 palabras	% sinónimos correctos	46,15	65,45
	% términos relacionados correctos	78,67	73,1
Lista 5.000 palabras	% sinónimos correctos	55	55,42
	% términos relacionados correctos	84,17	50,6
Lista 10.000 palabras	% sinónimos correctos	53,03	60,73
	% términos relacionados correctos	55,19	55,3

Figura 4.19: Tabla de resultados de la prueba cualitativa

```
[
  {
    "word": "inmueble"
  },
  {
    "synonyms": [
      "casa",
      "construcción",
      "edificio",
      "edificios"
    ],
    "offset": "spa-30-02913152-n"
  }
]
```

Figura 4.20: JSON devuelto al buscar los sinónimos fáciles de inmueble en el servicio web para la obtención de sinónimos fáciles

```
[  
  {  
    "word": "inmueble"  
  },  
  {  
    "synonyms": [  
      "casa",  
      "construcción",  
      "edificio",  
      "edificios"  
    ],  
    "offset": "spa-30-02913152-n"  
  }  
]
```

Figura 4.21: JSON devuelto al buscar los sinónimos fáciles de inmueble

```
GET http://127.0.0.1:8000/easyHyponym/json/word=inmueble&level=2

Response headers 5 Request headers 0

date: Sun, 05 May 2019 20:03:33 GMT
server: WSGIServer/0.2 CPython/3.7.0
content-type: application/json
x-frame-options: SAMEORIGIN
content-length: 318

[Array[2]
-0: {
    "word": "inmueble"
},
-1: {
    "offsetFather": "spa-30-02913152-n",
    "offset": "spa-30-80000135-n",
    "hyponyms": [Array[19]
        0: "arquitectura",
        1: "centro",
        2: "club",
        3: "residencia",
        4: "hotel",
        5: "casa",
        6: "casa",
        7: "biblioteca",
        8: "ministerio",
        9: "dependencia",
        10: "templo",
        11: "comedor",
        12: "restaurante",
        13: "pista",
        14: "colegio",
        15: "escuela",
        16: "torre",
        17: "templo",
        18: "teatro"
    ],
}
]
```

Figura 4.22: JSON devuelto al buscar los hipónimos fáciles de inmueble

```
GET http://127.0.0.1:8000/easyHyperonym/json/word=inmueble&level=2

Response headers 5
Request headers 0

date: Sun, 05 May 2019 20:12:35 GMT
server: WSGIServer/0.2 CPython/3.7.0
content-type: application/json
x-frame-options: SAMEORIGIN
content-length: 139

[  < > ]

[Array[2]
 -0: {
    "word": "inmueble"
 },
 -1: {
    "offsetFather": "spa-30-02913152-n",
    "offset": "spa-30-04341686-n",
    "hyperonyms": [Array[2]
      0: "construcción",
      1: "estructura"
    ],
 }
]
```

Figura 4.23: JSON devuelto al buscar los hiperónimos fáciles de inmueble

GET http://127.0.0.1:8000/metaphor/json/word=inmueble&level=2

Response headers 5 Request headers 0

date: Sun, 05 May 2019 20:23:39 GMT
server: WSGIServer/0.2 CPython/3.7.0
content-type: application/json
x-frame-options: SAMEORIGIN
content-length: 297

[Array[3]]

- 0: {
 "word": "inmueble"
},
- 1: {
 "offsetFather": "",
 "offset": "spa-30-02913152-n",
 "-metaphor": [Array[4]]
 0: "es una casa",
 1: "es una construcción",
 2: "es un edificio",
 3: "son unos edificios"
},
- 2: {
 "offsetFather": "spa-30-02913152-n",
 "offset": "spa-30-04341686-n",
 "-metaphor": [Array[2]]
 0: "es una construcción",
 1: "es una estructura"
},

Figura 4.24: JSON devuelto al buscar las metáforas de la palabra inmueble

```
GET http://127.0.0.1:8000/simil/json/word=inmueble&level=2

Response headers 5
Request headers (1)

date: Sun, 05 May 2019 20:47:31 GMT
server: WSGIServer/0.2 CPython/3.7.0
content-type: application/json
x-frame-options: SAMEORIGIN
content-length: 535

[Array[2]
-0: {
    "word": "inmueble"
},
-1: {
    "offsetFather": "spa-30-02913152-n",
    "offset": "spa-30-80000135-n",
    "simil": [Array[19]
        0: "es como una arquitectura",
        1: "es como un centro",
        2: "es como un club",
        3: "es como una residencia",
        4: "es como un hotel",
        5: "es como una casa",
        6: "es como una casa",
        7: "es como una biblioteca",
        8: "es como un ministerio",
        9: "es como una dependencia",
        10: "es como un templo",
        11: "es como una comedror",
        12: "es como una restaurante",
        13: "es como una pista",
        14: "es como un colegio",
        15: "es como una escuela",
        16: "es como una torre",
        17: "es como un templo",
        18: "es como un teatro"
    ],
}
],
```

Figura 4.25: JSON devuelto al buscar los símiles de la palabra inmueble



Figura 4.26: Interfaz de Aprende Fácil sin realizar ninguna búsqueda

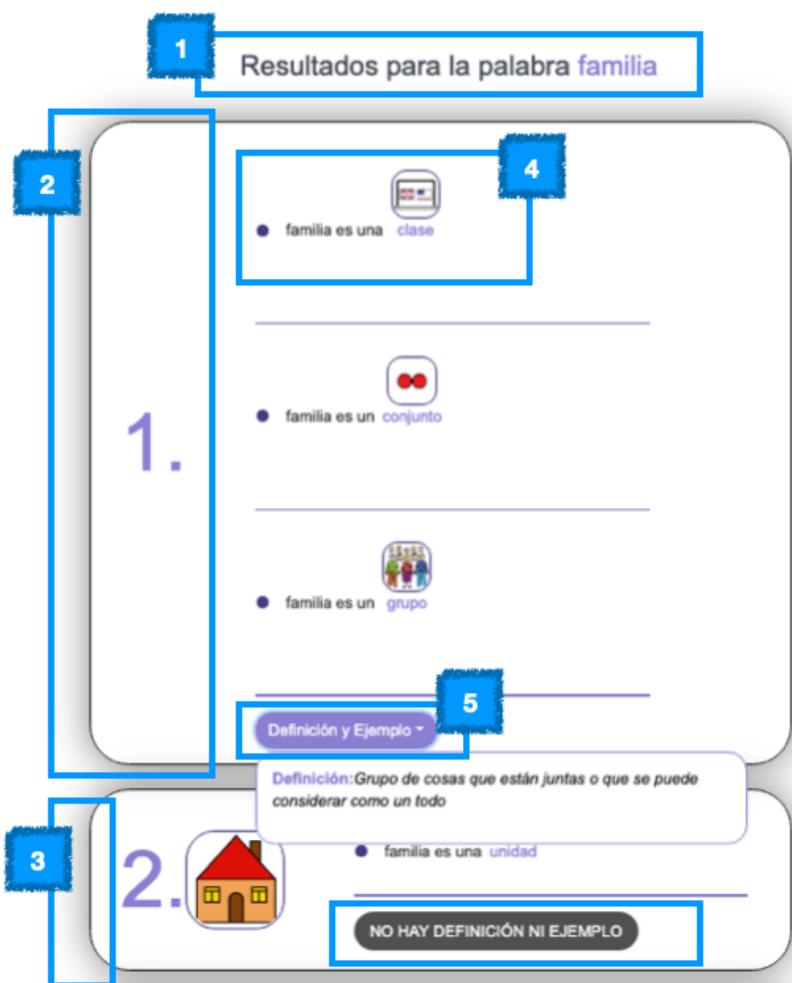


Figura 4.27: Interfaz de Aprende Fácil mostrando resultados

Capítulo 5

Trabajo Realizado

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

5.1. 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, integraron 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.

5.2. 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.

Capítulo **6**

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Chapter 6

Conclusions and Future Work

Conclusions and future lines of work.

Apéndice A

Artículos Periodísticos Utilizados para el análisis cualitativo y cuantitativo

En este apéndice se muestran los tres artículos periodísticos utilizados para realizar los análisis cuantitativos y cualitativos del diseño de la evaluación.

A.1. Artículo Tecnológico

Los fósiles de una escena primitiva, hallados con la precisión de una fotografía, demuestran que los seres humanos se comieron a los últimos grandes mamíferos que quedaban en América después de la última glaciación. La evidencia, que figura en un trabajo publicado por *Science Advances*, fue analizada por un equipo de arqueólogos argentinos del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) en la Universidad Nacional del Centro de la provincia de Buenos Aires (UNICEN) junto a investigadores estadounidenses.

El clima contra la depredación humana es (ahora se sabe) una falsa controversia científica respecto a la extinción de los megamamíferos en Sudamérica y en el Mundo. Las sucesivas evidencias han enfatizado una causa sobre la otra, pero en conjunto reflejan que ambas fueron determinantes en la desaparición de los grandes animales del Pleistoceno.

Se trata de un proceso que en América se inició en el deshielo y que el apetito humano probablemente sólo aceleró. “El clima jugó un rol también. Se extinguieron grandes animales en el mundo, no solamente acá aunque se extinguieron más en Sudamérica, también lo hicieron en Norteamérica y Europa. Entonces la discusión es: el clima y algo más. Este algo más creemos que son los seres humanos”, aclara el director de la investigación Gustavo

Politis, sentado en su oficina del área de Investigaciones Arqueológicas y Paleontológicas del Cuaternario Pampeano (INCUAPA), a pocos kilómetros del sitio del hallazgo. Sin embargo, agrega una advertencia para quienes cargan las culpas sobre los seres humanos. “En Sudamérica hay al menos 30 especies de megamamíferos que se han extinguido. Las que han sido cazadas son 5, 6 no más. No se puede explicar toda la extinción por la acción del hombre”. La caza no es el único daño que podríamos haber hecho en el pasado. “También puede haber pasado que los seres humanos hayan hecho disruptores en el ambiente como la introducción de nuevos parásitos o quemazones en los campos. Si el fuego produjo quemazón en poblaciones de animales con bajas tasas de reproducción, había un clima desfavorable y encima aparecieron seres humanos que los depredaron, los extinguieron” concluye Politis.

La imagen que este reciente descubrimiento permite evocar ocurre hace 12.600 años a la vera de un pantano (hoy convertido en arroyo) en la llanura pampeana argentina. Quedan pocos gigantes para cazar. Hace un poco más de frío, hay menos humedad que en la actualidad y el mar está más lejos. Sobre los pastizales se erige quejoso un perezoso de cuatro metros de alto (como la estatura del oso y el madroño) y unas tres toneladas (el equivalente a cinco toros de lidia). Un grupo de cazadores armados con lanzas y proyectiles de piedra lo ataca. El grotesco animal, cuyo nombre científico es Megaterio, no logra ganar la batalla y acaba convertido en cena, abrigo y herramientas. Para saber esto, los investigadores desenterraron y examinaron fósiles durante 18 años en un campo de hacienda al que llegaron gracias a la suerte de un agricultor que, operado de la cadera, tuvo que dejar un tiempo el caballo y recorrer el campo andando. La caminata lenta y larga lo sorprendió con un fémur inmenso y ennegrecido asomando de la tierra.

Esa oscuridad se convertiría en la clave para desentrañar la prehistórica escena. “Es un ambiente con mucha materia orgánica porque era un antiguo pantano”, describe Politis. En esas condiciones, explica, es muy probable que se produzca la reacción de Maillard sobre el fósil; cuando las moléculas de colágeno -que permiten conocer la antigüedad de un hueso- se juntan con las de los ácidos húmicos y fúlvicos -esos que forman el humus, la tierra negra fértil que aman los jardineros. “Para separar eso hay que hacer un tratamiento especial porque si no, se datan las dos cosas juntas”, advierte Politis, licenciado en Antropología y doctor en Ciencias Naturales. “Digamos que te rejuvenece la edad del fósil”, agrega en la misma oficina, y con iguales títulos académicos, el segundo autor del trabajo Pablo Messineo. “El problema que teníamos siempre era que la datación nos daba entre 7.500 y 8.000 años atrás. Es una edad muy reciente para lo que es la extinción de los megamamíferos en Sudamérica, estimada en 12.000 años”. Hasta que llegaron a Thomas Stafford y su laboratorio. Este investigador desarrolló a principios de los '80 un método único que permite separar el colágeno del resto de

la materia orgánica. “Al separar el colágeno y datarlo te asegurás de que esa contaminación no exista más. Lo que hicimos ahora fue separarlos y datar ambos. Entonces, el colágeno nos dio 12.600 años y lo demás, 9.000. Así confirmamos que los ácidos estaban contaminando la muestra”, explica Messineo.

El paso del tiempo, el clima, la erosión, pueden tergiversar la historia. “Si hay varios eventos a lo largo del tiempo, se pueden mezclar y es difícil separarlos. Acá hay uno solo; no pasó nada antes ni después. Tenemos una foto arqueológica de ese momento. Hay pocas cosas pero muy coherentes entre sí”, asegura el director del equipo de investigación. “Una de las más interesantes es un artefacto que es una especie de cuchillo de piedra que se les rompió y tenemos los dos pedazos. Uno lo encontramos en 2003 y otro ahora. Estaban a 3 metros. Cuando los juntamos nos dimos cuenta de que uno lo habían tirado cuando se les rompió y al otro lo siguieron reactivando, tiene el filo más usado”, destaca Messineo.

“Es una conducta esperable; gente que está carneando, cuando el filo se les agota lo reactivan, cuando lo reactivan se puede romper y con el pedazo que les conviene más, siguen cortando. Es como llegar a una escena ni bien ha ocurrido. Están los huesos, las herramientas, solo falta la gente”, detalla con fascinación Politis, distinguido por su trayectoria por el Estado argentino. La evidencia de que los humanos se comieron al perezoso gigante es contundente. “Encontramos marcas de corte en una costilla, eso indica que sin dudas que los grupos humanos lo carnearon. Después encontramos dos instrumentos hechos con las costillas del megaterio. Es otro indicio de que no es solamente la asociación de los artefactos de piedra con los huesos sino que los tipos estuvieron ahí procesando al animal. Eso hoy es indiscutible”, afirma.

Las dataciones de este estudio, en el que también trabajó Emily Lindsey, investigadora del Museo Rancho La Brea de Los Ángeles, Estados Unidos, permiten saber que los seres humanos y los grandes mamíferos coexistieron durante unos 1.500 años y arrojan sospechas sobre otras investigaciones. “Hay que re-datar otros hallazgos también. Ahora sospechamos de las dataciones de los otros sitios porque estaban hechas con métodos más convencionales”, advierte Politis. Si el presupuesto (por el momento suspendido) lo permite, esas nuevas dataciones podrán descontaminarse en laboratorios propios ya que la UNICEN y el CONICET están trabajando en el desarrollo local de esa tecnología disponible, por ahora, en muy pocos laboratorios de Estados Unidos y Europa.

A.2. Artículo Deportivo

Las tres caídas consecutivas en los cuartos de final de la Champions habían dejado tocado al Barcelona, sobre todo la temporada pasada, cuando

los muchachos de Valverde se derrumbaron por 3-0 ante la Roma. Los directivos apuntaron al técnico, actitud que alertó al vestuario. El grupo, en la intimidad, había despojado de cualquier responsabilidad a su entrenador, un tipo al que los jugadores reconocen y respetan, esencialmente por su mano izquierda para gestionar el camerino. Sabían que se habían quedado sin gasolina. Luis Suárez, uno de los pesos pesados, reconoció públicamente que se había equivocado ante el Leganés, en la previa de la debacle azulgrana en Roma. La herida se acentuó cuando el Madrid levantó su tercera Champions seguida. Messi no lo toleró: “Queremos que vuelva al Camp Nou la copa linda y deseada”.

Reconocido el error, para esta campaña la consiga era clara: había que confiar más en las rotaciones del Valverde. Y esta vez, cuando llegaron los cuartos de la Champions ante el United, nadie protestó. El técnico borró a todas sus estrellas del duelo frente al Huesca en la víspera de la vuelta ante el equipo de Manchester. Funcionó. Los suplentes empataron en El Alcoraz y los titulares barrieron al conjunto de Ole Gunnar Solskjær. Sin embargo, en la previa de las semifinales ante el Liverpool, Valverde apostó por los mejores contra la Real y viajarán todos, salvo Rakitic, a Vitoria. El Barça quiere llegar a la eliminatoria contra el equipo de Klopp, el martes 1 de mayo en el Camp Nou, con la Liga en el Museo del Camp Nou. El problema es que no es lo mismo salir campeón el miércoles —necesita ganar hoy al Alavés (21.30, beIN LaLiga) y que mañana pierda el Atlético con el Valencia— que el sábado, día en que el Levante visita el Camp Nou, antes del encuentro de Champions. ¿Qué alineación sacará el técnico ante el equipo granota si se juega el título y el miércoles aguarda el Liverpool?

“Nos ponemos estupendos para ver cuándo vamos a ganar... Lo que queremos es ganar cuanto antes, pero la gente lo ve desde fuera y piensa que todo es sencillo”, resolvió Valverde. “Cuando se gana la Liga al final parece que la alegría es superior, pero cuando tienes una ventaja sobre el segundo parece un título previsible”, completó el Txingurri. “Ojalá pudiéramos ganar siempre con margen”.

Valverde dejó caer que habrá rotaciones en Vitoria. “Desde luego, es muy posible que haya cambios”, advirtió. El técnico lleva perfectamente la cuenta de los minutos de sus jugadores, y los titulares están más descansados que la última campaña. “Físicamente los veo bien”, subrayó; “es normal que a estas alturas de la temporada los equipos acusemos el paso del tiempo, pero nos alimentamos de las sensaciones que tenemos. Los retos eliminan cualquier cansancio, cuando hay objetivos las piernas están frescas”. Valverde confía en la ambición de sus futbolistas, pero, por las dudas, los quiere descansados ante el Liverpool.

A.3. Artículo Político

El PP ha criticado este martes los ataques recibidos por el líder de Ciudadanos durante el debate electoral. “Rivera le hizo el trabajo sucio a Sánchez”, se ha quejado el secretario general de los populares, Teodoro García Egea. El candidato de Cs salió al choque tanto contra Pedro Sánchez como contra Casado, mientras este evitaba el enfrentamiento con él. Solo una vez –cuando Rivera sugirió que podían pactar con el PNV como ya habían hecho en el pasado– entró Casado en el cuerpo a cuerpo: “Ni sus votantes ni los míos entienden lo que está diciendo, Usted no es mi adversario. Yo no voy a pactar ni con el PNV ni con Sánchez. En materia de pactos soy más creíble que usted”.

Todos los partidos se arrojan este martes la victoria del debate en TVE, a pocas horas del segundo asalto en Atresmedia. Fuentes del PP indican que Casado hizo “el debate que quería hacer”, con un “tono moderado y presidenciable”. El candidato popular quiere tratar de recuperar un perfil centrado en la última semana de campaña y dejar atrás las graves acusaciones y descalificativos empleados contra Sánchez, como cuando le acusó de preferir “las manos manchadas de sangre”. Los populares dudan sobre la estrategia. Algunos dirigentes consultados por EL PAÍS creen que con el tono moderado del lunes, Casado puede decantar a algunos indecisos que se debaten entre votar al PP o a Ciudadanos. Otros señalan que el líder estuvo excesivamente plano y dejó que Rivera acaparara protagonismo. Fuentes de la dirección indican, en cualquier caso, que el formato de esta noche, con preguntas y repreguntas, da pie a intervenciones algo más atrevidas.

“Rivera todavía no se ha enterado de que el enemigo se llama Pedro Sánchez”, ha subrayado este martes el secretario general del PP. Los populares quieren unir fuerzas (y votos) en torno al enemigo común, ya que su única oportunidad de gobernar es sumando los escaños de lo que el presidente llamó el lunes, insistentemente, “las derechas”, incluyendo a Vox.

Pero Rivera salió a la cita a proyectarse como alternativa a Sánchez y eso implicaba también tratar de noquear a Casado, con el que se disputa el liderazgo del bloque de centroderecha. Los suyos están convencidos de que lo logró, ante un Casado demasiado contenido.

“Todos los españoles vieron ayer un ganador claro, el único capaz de demostrar que es una alternativa a Pedro Sánchez, el único que lleva mucho años luchando por la igualdad de todos los españoles, el único con la fuerza y la valentía necesarias para hacer frente a los grandes retos de España”, ha enfatizado hoy sobre Rivera Inés Arrimadas, candidata de Cs por Barcelona. La dirección de Ciudadanos considera que el de ayer fue un “punto de inflexión” en la campaña electoral en el que su candidato salió beneficiado. Rivera, según fuentes de la cúpula, volverá a salir hoy a la ofensiva y a intentar liderar el segundo debate.

En la izquierda, el PSOE ha cargado contra los “exabruptos” de la derecha y ha reivindicado que el candidato socialista fue el único en hacer un debate en positivo. “Creo que el tono de los candidatos de derechas en el debate fue el mismo tono que hemos estado viendo durante toda la campaña: uno donde han primado más los insultos, los exabruptos, con pocas propuestas y poco proyecto de país”, ha defendido la candidata del PSOE por Barcelona, Meritxell Batet. Sánchez puso sobre la mesa, a juicio de los socialistas, “un proyecto de país, muchas propuestas e hizo un listado exhaustivo” de la labor del Gobierno socialista en sus diez meses al frente del Ejecutivo.

Por su parte, Unidas Podemos ha destacado que el candidato del PSOE no contestó ayer a las reiteradas preguntas de Pablo Iglesias sobre si pactará o no con Ciudadanos tras el 28 de abril, lo que para la formación morada es revelador de que esa es la intención de los socialistas. El secretario de Organización de Podemos, Pablo Echenique, se ha dirigido este martes a los indecisos y a los votantes socialistas para advertirles de que su voto el 28 de abril puede acabar “en un Gobierno de derechas en el que Albert Rivera sea ministro”.

Ese es el punto central de la campaña de Podemos, el de alertar sobre un posible pacto del PSOE con Ciudadanos, aunque la ejecutiva del partido de Albert Rivera ha aprobado cerrar la puerta a ese acuerdo, como el candidato naranja recordó también ayer durante el debate electoral. Su apuesta es un pacto con el PP para el que ayer volvió a tender la mano a Casado, sin respuesta. En el segundo asalto, el debate de esta noche, los pactos poselectorales volverán a dar previsiblemente momentos de choque entre los candidatos.

Apéndice **B**

Título

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el celebro 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.

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

MANNING, C. D. y SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. ISBN 0-262-13360-1.

MORENO ORTIZ, A. *Diseño e implementación de un lexicón computacional para lexicografía y traducción automática*. Facultad de Filosofía y Letras., 2000.

NOMURA, M., SKAT NIELSEN, G. y TRONBACKE, B. *Directrices para materiales de lectura fácil*. Federación Internacional de Asociaciones e Instituciones Bibliotecarias, 2010.

QUILLIAN, M. R. *Semantic Memory*. Cambridge, 1968.

SHNEIDERMAN, B. y PLAISANT, C. *Diseño de interfaces de usuario : estrategias para una interacción persona-computadora efectiva*. Pearson Education, 2006.

- SOWA, J. *Conceptual structures: Information processing in mind and machine.* Addison-Wesley Longman Publishing Co., Inc., 1983. ISBN 0-201-14472-7.
- TORRES, J. *Servicios Web.* Universidad Complutense de Madrid, 2017.
- VEALE, T. y LI, G. *Exploding the Creativity Myth: The Computational Foundations of Linguistic Creativity.* Bloomsbury Academic, 2012.
- VEALE, T. y LI, G. *Creating Similarity: Lateral Thinking for Vertical Similarity Judgment.* In Proceedings of ACL 2013, the 51st Annual Meeting of the Association for Computational Linguistics, 2013.
- VEGA LEBRÚN, C. A. *Integración de herramientas de tecnologías de información como soporte en la administración del conocimiento.* Trabajo doctorado, Universidad Popular Autónoma del Estado de Puebla, 2005.

*-¿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*

