Mejora de la Comprensión Lectora para la Inclusión mediante Figuras Retóricas

Trabajo de Fin de Grado
Curso 2018–2019

Autores
Irene Martín Berlanga
Pablo García Hernández

Directores
Virginia Francisco Gilmartín
Gonzalo Méndez Pozo

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

Mejora de la Comprensión Lectora para la Inclusión mediante Figuras Retóricas

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

Autores
Irene Martín Berlanga
Pablo García Hernández

Directores
Virginia Francisco Gilmartín
Gonzalo Méndez Pozo

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

31 de mayo de 2019

Agradecimientos

Queremos agradecer el apoyo recibido por parte de nuestros directores Virginia y Gonzalo, por su paciencia y sus ánimos cuando más lo necesitábamos.

Agradecer también a nuestros amigos, familiares y a nuestras parejas, por estar siempre a nuestro lado, por aguantar nuestros malos días, por asesorarnos para mejorar este trabajo. No tenemos suficientes palabras de agradecimiento para expresar todo lo que sentimos.

Tenemos muchísima suerte de teneros.

Resumen

Actualmente, en España se calcula que aproximadamente el 1 % de la población sufre algún tipo de discapacidad intelectual. Para estas personas, la realización de tareas tan sencillas y cotidianas como ir a comprar o leer un libro pueden resultar muy complicadas, ya que tienen dificultad de comprensión para comprender muchos de los términos empleados en el día a día. Una posible solución para entender un concepto es la consulta del mismo en un diccionario. Pero esta no es una solución viable para estos colectivos, ya que las definiciones ofrecidas suelen tener un lenguaje complicado y difícil de entender para ellos.

Para ayudar a estas personas con sus problemas de comprensión, se ha implementado una aplicación web que dada una palabra la define en base a otras palabras relacionadas con ella más fáciles y que el usuario conoce. Para dar estas definiciones se emplearon metáforas y símiles. Al tratarse de una aplicación web, está al alcance de cualquier usuario que disponga de un dispositivo con conexión a internet.

Para diseñar la aplicación se ha hecho un diseño centrado en el usuario, contando con las opiniones de expertos. Se quiere conseguir una aplicación que se adapte a las necesidades reales del colectivo objetivo.

La aplicación está compuesta por varios servicios web, para obtener tanto las palabras relacionadas como para generar los símiles y las metáforas. Además, para facilitar la comprensión del usuario, se han implementado varios servicios web complementarios. Uno que obtiene el pictograma asociado a una palabra, y otro que proporciona una descripción y un ejemplo.

La aplicación ha sido evaluada por usuario finales. De esta evaluación se pudo concluir que el trabajo es útil para personas cuyo grado de discapacidad no fuese muy elevado, y que queda trabajo por hacer si queremos que la aplicación sea útil para personas con un grado de discapacidad mayor.

Palabras clave

Discapacidad cognitiva, Inclusión, Palabras fáciles, Palabras complejas, Figuras Retóricas, Metáfora, Símil, Servicios Web, Diseño centrado en el usuario, Pictogramas

Abstract

Today, 1% of the Spanish population suffers from some kind of cognitive disability. It is very difficult for these people to perform simple tasks such as reading a book or doing the daily shopping, due to their difficulties to understand common concepts. The easiest solution would be to look up for words in a dictionary, but this solution is not appropriate for these people because the definitions and concepts used in them include complex concepts and they are also hard to understand.

In order to help these people with their inability to understand certain words, we have developed a web application in which they can look up a word and obtain a definition relating these difficult words with easier ones they are likely to know and understand. This has been accomplished by using similes and metaphors related to those words. The fact that it is a web application makes it more handy for everyone with an Internet connection.

We have developed the application through a User-centered design, taking into account the opinions and advices of experts. The aim of this approach has been to develop an application adapted to the real needs of the target users.

This application is composed of a few web services that can obtain related words, similes and metaphors. Also, in order to make it easier to use for the user, we have implemented two more web services: one of them finds pictograms and the other one finds examples and a description of the word they look for.

The application has been tested by users with cognitive impairments. After being tested by these users, we have concluded that this application is useful for people who have mild disabilities but further adaptations have to be carried out in order to make it useful for people with medium or severe impairments.

Keywords

Cognitive disability, Inclusion, Easy words, Complex words, Rhetorical figures, Metaphor, Simile, Web service, User-centered design, Pictograms

Índice

Índice de figuras

Índice de Listados

Chapter 1

Introduction

1.1. Motivation

Nowadays, Spanish is the second most spoken language around the world and it is comprised of more than 90.000 words. It is a language with many terms, and depending on the context in which they are found, they can have multiple meanings. For example, the word "*gato*" can refer to an animal or a tool to lift a car. If this is hard for any person, for certain groups of people affected by some cognitive disorder it is even more so, affecting them in their daily, professional or personal life. For example, the simple action of reading a newspaper is very difficult for them since they do not know what many words mean. Another common example could be to read an instruction manual, when they are in the same situation of not being able to understand certain concepts.

One of the solutions that could be thought of at first is to look up the meaning of the words in a dictionary. However, this solution does not help them, since the definitions that appear in many cases do not use easy terms or phrases. For example, the word computer has the following definition in the dictionary: "*Electronic machine that, through certain programs, can store and process information, and solve problems of various kinds*". This definition can be quite complicated to understand by a person with cognitive disabilities, since it is a long sentence and it uses technical terms.

The solution that has been thought to solve this problem is to offer a definition that compares the difficult concept with other easier concepts already known by the user. To make these comparisons we will use rhetorical figures, more specifically we will make use of metaphors, analogies and similes. For example, for the word "vehicle" the following results could be obtained:

- Metaphor: *A vehicle is a car.*

- Analogy: *A vehicle is like a machine.*
- Simile: *A vehicle is fast like a plane.*

We think that the use of rhetorical figures will make it easier to understand difficult concepts. In these definitions, short sentences and simple concepts will be used. In addition, we will add pictograms to the results to reach a broader collective of users.

Therefore, what we propose is to create an application that, when a complex word is searched, returns a definition of that word formed by comparisons with simpler concepts making use of rhetorical figures.

1.2. Goals

The main goal of this project is to create a web application based on web services that, when the user searches a complex word, returns a definition of that word comparing it with other simpler words using similes, analogies or metaphors.

In order to obtain these definitions, it will be necessary to study how to obtain the terms related to a concept, as well as to make out which of these results are easy words and which are difficult words. It will also be necessary to know what kind of rhetorical figures exist and what to use according to the relation between the initial concept and the easy concept. Ihis way, the final user can get a correct and satisfactory result.

The application will be built with web services that provide functionality to the application and that are reusable in other applications, making it adaptable to the needs of the users. The developed web services will be available in a public API so that it can be used by other developers to integrate our services in their applications, thus making them easier and faster to build. The application will be built incrementally, adding value to the product little by little. In this way, we can test the different working hypotheses progressively and make modifications in a simple way to get an application that meets the needs of the users.

The design of the interface will be centered on the user, and for this, as much information as possible about the end user must be obtained. In this way, a design is made based on the users needs. Since this work is focused on people with cognitive disabilities, a design that adapts even more to their needs and limitations should be made.

Finally, we must not forget the academic objectives of this work, such as putting into practice the knowledge acquired during the degree and expanding our knowledge in different areas.

By achieving the objectives described above, we will obtain a quality product, with great social and academic utility, that can help many users to learn new concepts in a simpler way.

1.3. Project management methodology

Since the beginning of the project, efficiency and continuous improvement have been sought, which is why meetings have been held with the directors of this work, every two or three weeks. In these meetings the work done since the last meeting was reviewed. We looked for solutions to the problems and doubts that could arise and we fixed the tasks to be carried out until the next meeting. On the other hand, there has been constant communication with both directors via email to ask questions. In relation to configuration management, the online development platform GitHub has been used to keep track of project versions. In addition, a task manager that has served as an information radiator has been used and all project members have been able to know at any time the status of the project and each of its tasks. For this, Trello has been chosen, since it has a simple, friendly interface that does not lead to confusion when creating new tasks.

There are two types of tasks on our board: those related to code and those related to memory. A distinction has been made between both, since the way to change their status on the board varies significantly depending on the type of task. In order to make this distinction in the tasks, the word CODIGO or the word MEMORIA has been written as it corresponds in front of the description. Our task board has three columns:

- TO DO: Tasks to be carried out, stripped to the greatest possible detail and trying to make them as independent as possible from each other. In this way, it is ensured that each member of the team works on a specific task that does not influence the work of the other partner.
- In process: At the moment in which a member of the team is assigned a task, it is passed from the TO DO column to the column "In process", which indicates that it is in the process of develop and no other partner can be to work on it.
- In review: In this column are the tasks completed but not validated. The validation depends on the type of task: if the task is of type code, the validation must be done by the team members, performing tests to verify that it really fulfills its objective and does not cause any error, and if it is of the memory type the validation must do the directors.
- Done: In this column are the tasks already validated by the directors or by the members of the team depending on the type of task.

1.4. Memory structure

In **Chapter 2**, the state of the art is presented, in which the semantic networks will be explained: what they are, their different types and some

tools that implement them. Afterwards, what it is the easy reading and how it is applied will be explained, and the rhetorical figures will be discussed. Finally, we will explain what web services are, the existing types, their architecture, and the advantages and disadvantages of their use.

In **Chapter 3**, the tools used for the creation of this work will be explained, such as Django for the development of the application and SpaCy for the labeling of words. We will describe what these tools are used for and their characteristics.

In **Chapter 4**, we will explain everything related to the application: its design, the implementation, how the database is designed, what semantic network was used, the backend and the frontend. Finally, the evaluation carried out with users will be described.

In **chapter 5**, we describe the work carried out by each of the authors of this project.

In **chapter 6**, we detail the conclusions obtained after finishing the project as well as the future work that could be carried out.

Capítulo 1

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 alguna persona con discapacidad cognitiva ya que utiliza términos más técnicos y es una frase bastante larga. La 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 por el usuario. Para hacer estas comparaciones usaremos las figuras retóricas, más concretamente haremos uso de metáforas, analogías y símiles. Por ejemplo,

[1] https://dle.rae.es/?id=A4hIGQC

para la palabra vehículo se podrían obtener los siguientes resultados:

- Metáfora: *Un vehículo es un coche.*
- Analogía: *Un vehículo es como una máquina.*
- Símil: *Un vehículo es rápido como un avión.*

Creemos que el uso de las figuras retóricas facilitará 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 una definición de la palabra mediante comparaciones con conceptos más sencillos haciendo uso de 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 con un concepto, así como distinguir cuales de estos resultados son palabras fáciles y cuales son palabras 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 probar 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 a 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 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[2].

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 dependiendo del tipo de tarea. 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.

[2]https://github.com/NILGroup/TFG-1819-Analogias

- 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 validación 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.

1.4. Estructura de la memoria

En el **capítulo dos** se presenta el Estado de la Cuestión, en el que se explicarán las redes semánticas: que son, que tipos existen y algunas herramientas que las implementan. Después, se explicará que es la lectura fácil y como se aplica y se hablará de las figuras retóricas. Por último se explicarán que son los servicios web, los tipos que existen, 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. Se describirá para que se utilizan estas herramientas y sus características principales.

En el **capítulo cuatro** se explicará todo lo relacionado con la aplicación. Su diseño, la implementación donde se explicará su arquitectura, como está diseñada la base de datos, que red semántica se utilizó, el backend y el frontend. Por último se describirá la evaluación realizada con expertos.

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

En el **capítulo seis** se detallan las conclusiones obtenidas tras la finalización del proyecto así como el trabajo futuro que se podría realizar.

Capítulo 2

Estado de la Cuestión

Como se ha comentado en el capítulo anterior, el objetivo principal de este Trabajo de Fin de Grado es construir una aplicación que ayude a personas con discapacidad cognitiva a entender palabras más complejas mediante comparaciones con 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 han empleado redes semánticas que son una de las principales técnicas de representación del conocimiento empleada en el Procesamiento del Lenguaje Natural (PLN). En la sección 2.1 se explicará en qué consiste el PLN, las redes semánticas que existen y las distintas herramientas disponibles para representar conocimiento que nos pueda ayudar a buscar conceptos relacionados con otro concepto dado.

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 fáciles 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 capítulo anterior, 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. Redes Semánticas

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

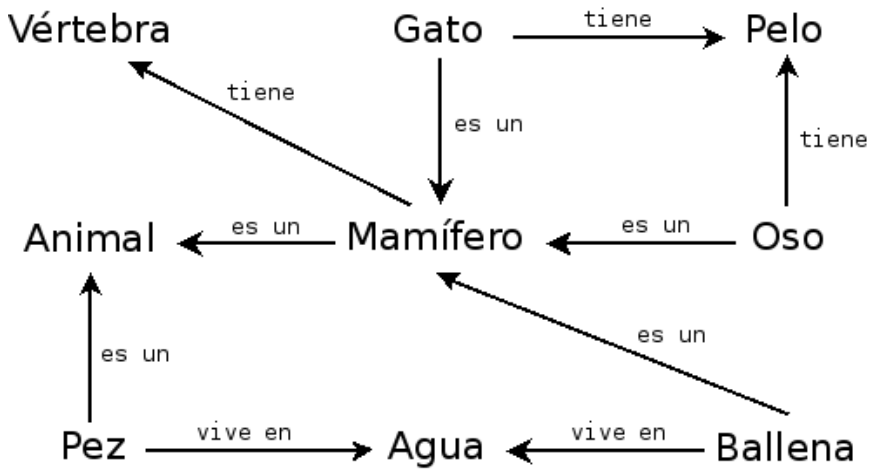

Figura 2.1: Ejemplo de Red Semántica

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 cumplirlo.

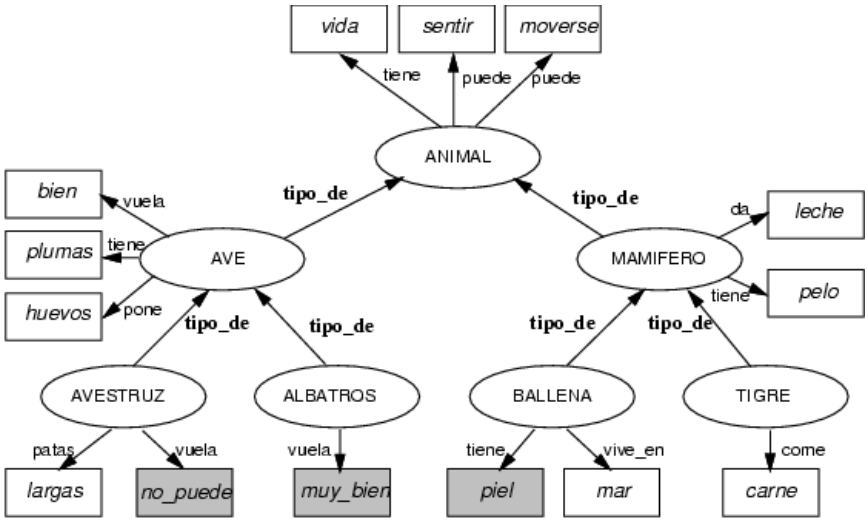

Figura 2.2: Ejemplo de Red de Marco

- 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.

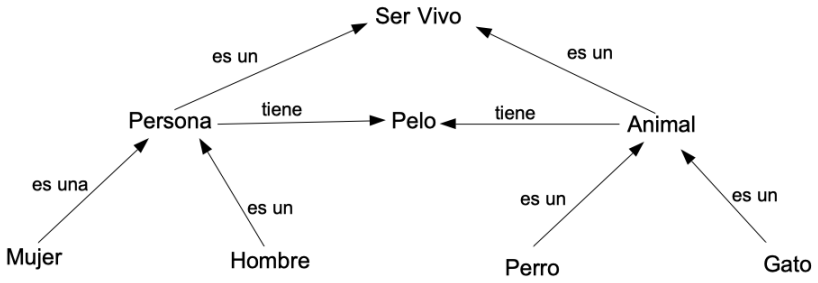

Figura 2.3: Ejemplo de Red de IS-A

- Grafos Conceptuales: existen dos tipos de nodos en estas redes: nodos de conceptos, los cuáles representan una entidad, un estado o un proceso y los nodos de relaciones, que indican como se relacionan los nodos de concepto. En este tipo de red semántica no existen enlaces entre los nodos con una etiqueta, sino que son los propios nodos los que tienen el significado. 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.

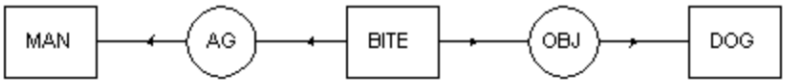

Figura 2.4: Ejemplo de Grafo Conceptual

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 etimologicamente relacionados, símbolos, etc.

ConceptNet dispone de una aplicación web[1] donde se pueden buscar palabras 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 también en distintos idiomas (saco o chaquetear), 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 letra que indica si es un verbo (v), un sustantivo (n) o un adjetivo (a).

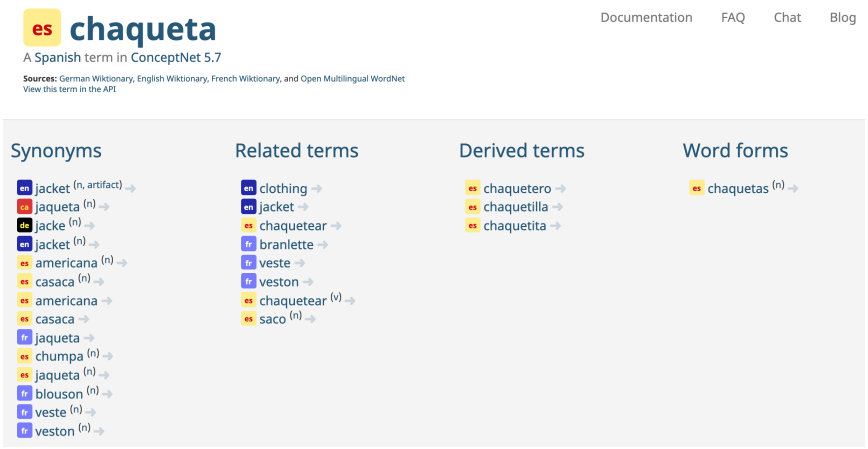

Figura 2.5: Resultados de ConcepNet para la palabra chaqueta

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, ConceptNet dispone de un servicio web[2] 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. El JSON devuelto en este caso consta de cuatro campos principales[3]:

- @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.

[1] http://conceptnet.io/
[2] http://api.conceptnet.io
[3] https://github.com/commonsense/conceptnet5/wiki/AP

- @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

Thesaurus[4] 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.

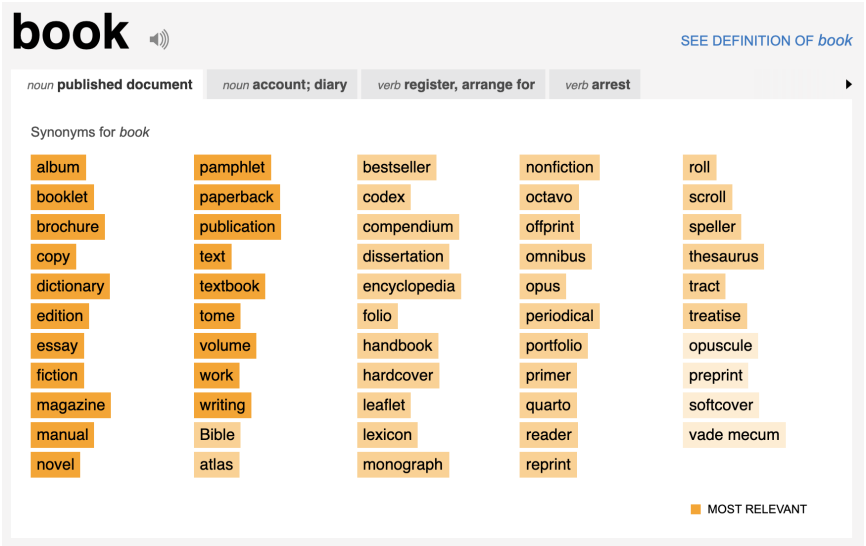

Figura 2.6: Resultados de ConcepNet para la palabra polisémica *book*

Por otro lado, esta aplicación proporciona una API[5] 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 aparcen los sinónimos. En caso de que alguna palabra sea un antónimo aparecera 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 Rex[6] 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.

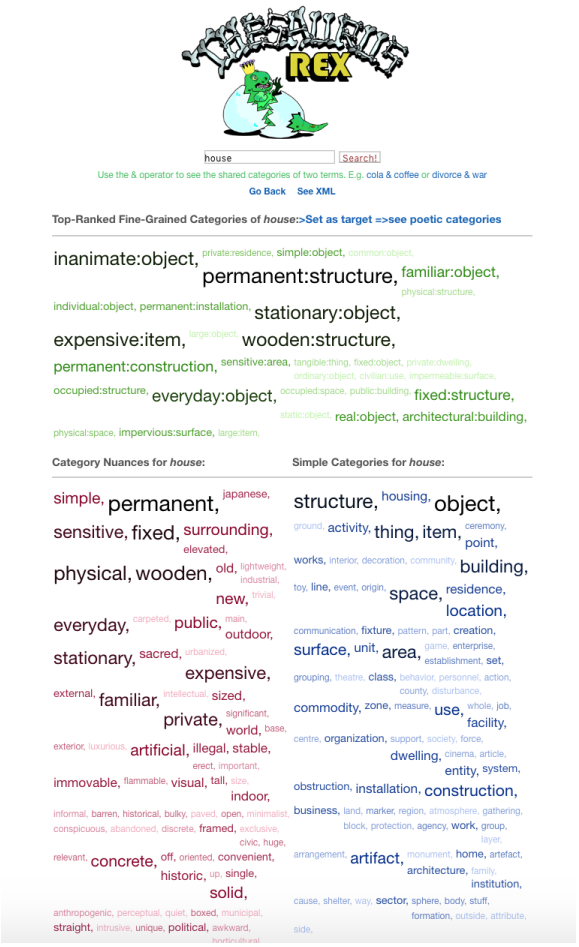

Figura 2.7: Resultados búsqueda Thesaurus Rex con la palabra *house*

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*. 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*. Todos los resultados de las tres categorías tienen un peso *(weight)* asignado, esto significa que cuanto mayor sea el peso mayor es la similitud con el concepto dado.

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 web[7], 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. Estos pueden ser bigramas, tigramas, etc., según se trate de dos, tres, etc. términos consecutivos. Metaphor Magnet lo que hace es buscar en la Web cuantas veces aparece el concepto introducido junto con otra palabra en un n-grama. De esta forma, se obtienen los conceptos más utilizados junto con el concepto introducido. 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[8] 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 *<Text>* que contiene la metáfora, por ejemplo *"protecting:home"*. Por último, una etiqueta *<Score>* que muestra un número, cuanto mayor sea este número más relacionado estará la metáfora 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*.

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>
```

[7] http://ngrams.ucd.ie/metaphor-magnet-acl/

[8] http://ngrams.ucd.ie/metaphor-magnet-acl/q?kw=house&xml=true

```
<Source Name="house">
<Text> charming:mansion </Text>
<Score> 83 </Score>
</Source>
<Source Name="house">
<Text> strong:rock </Text>
<Score> 80 </Score>
</Source>
<Source Name="house">
<Text> strong:elephant </Text>
<Score> 80 </Score>
</Source>
<Source Name="house">
<Text> 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>
```

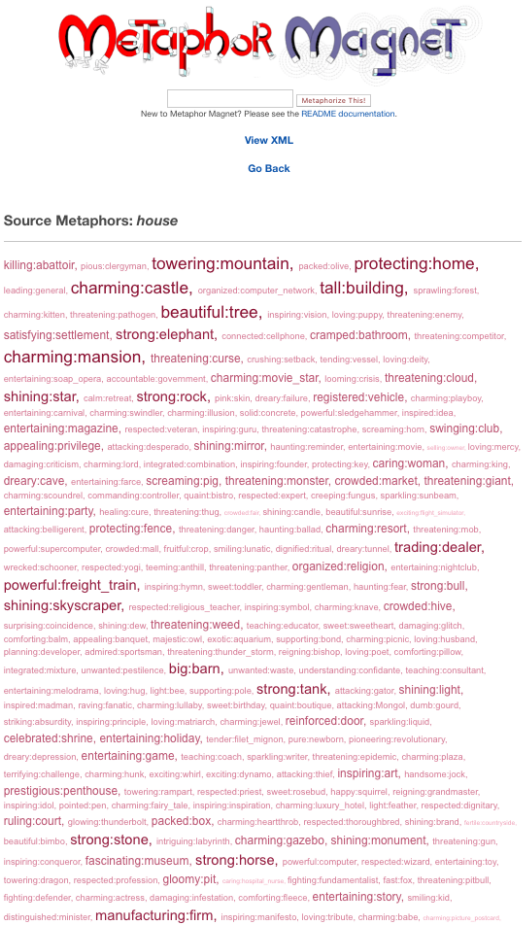

Figura 2.8: Resultados búsqueda Metaphor Magnet con la palabra *house*

2.1.5. WordNet

WordNet es un *corpus*[9] perteneciente a NLTK *(Natural Language Toolkit)*[10] 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 otras[11]. 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 otra[12]. 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.

[9] Colección de documentos de texto
[10] Conjunto de bibliotecas y programas para el Procesamiento del Lenguaje Natural
[11] https://dle.rae.es/?id=KRW1qe2
[12] https://dle.rae.es/?id=KU5UAn5

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 fácil[13] 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 fácil[14]. 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 IFLA[15] y de Inclusion Europe[16] 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.

[13] https://www.discapnet.es/areas-tematicas/diseno-para-todos/accesibilidad-de-comunicacion/lectura-facil

[14] https://dilofacil.wordpress.com/2013/12/04/el-origen-de-la-lectura-facil/

[15] International Federation of Library Associations and Institutions

[16] Una 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 corpus[17] 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.

[17] 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)[18],*"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 muy preciado 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 mas 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

[18] https://dle.rae.es/?id=WISC3uX

en el cual estén programadas o ejecutadas en cualquier tipo de plataforma[19]. Según el W3C (*World Wide Web Consortium*)[20],*"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 encontrarle. 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 remotos[21]. 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.

[19] http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html
[20] https://www.w3.org/
[21] 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-pqff98fe8j7d</m:
                reference>
            <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
        </m:reservation>
        <n:passenger xmlns:n="http://mycompany.example.com/employees"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/
                next">
            <n:name>Fred Bloggs</n:name>
        </n:passenger>
    </env:Header>
    <env:Body>
        <p:itinerary xmlns:p="http://travelcompany.example.org/reservation
            /travel">
            <p:departure>
                <p:departing>New York</p:departing>
                <p:arriving>Los Angeles</p:arriving>
                <p:departureDate>2007-12-14</p:departureDate>
                <p:departureTime>late afternoon</p:departureTime>
                <p:seatPreference>aisle</p:seatPreference>
            </p:departure>
            <p:return>
                <p:departing>Los Angeles</p:departing>
                <p:arriving>New York</p:arriving>
                <p:departureDate>2007-12-20</p:departureDate>
                <p:departureTime>mid-morning</p:departureTime>
                <p:seatPreference></p:seatPreference>
            </p:return>
        </p:itinerary>
    </env:Body>
</env:Envelope>
```

- Servicios Web RESTful: RESTful es un protocolo que suele integrar mejor con HTTP que los servicios basado en SOAP, ya que no requieren mensajes XML. Cada petición del cliente debe contener toda la información necesaria para entender la petición, y no puede aprovecharse de ningún contexto almacenado en el servidor.

2.4.2. 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 capas[22]:

- 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 que operaciones soporta y como 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.

[22] 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 desventajas[23]:

- 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.

[23] 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.

3.1. Django

Para construir un servicio web se necesita una manera de gestionar los elementos propios de estos servicios[1]. Algunos de estos elementos son el procesamiento de formularios y el mapeo de urls, que requieren *frameworks* web[2] para su gestión. Django es un *framework* de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles y se basa en el patrón MVC[3]. 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 siguientes[4]:

- Seguridad: Implementa por defecto algunas medidas de seguridad para evitar SQL Injection (adición de consultas SQL malignas que puedan alterar la base de datos) o Clickjacking (que el usuario haga click en

[1] https://tutorial.djangogirls.org/es/django/

[2] estructuras que contienen los componentes necesarios para el desarrollo de aplicaciones web

[3] https://docs.djangoproject.com/en/2.0/

[4] https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/

un enlace oculto, para obtener información del mismo sin su permiso o incluso tomar el control de su ordenador).

- 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[5] (Object Relational Mapper). Django tiene su propio ORM, con el que se pueden hacer consultas de manera muy intuitiva.

- Popularidad: Django es muy popular, por lo que hay mucha documentación disponible y muchos hilos en foros de programación en donde encontrar soluciones para cualquier problema que surja.

3.2. SpaCy

Dentro de las palabras relacionadas con el concepto introducido por el usuario, solo nos interesan los nombres, verbos y adjetivos. Para identificar estas categorías léxicas se necesita un analizador léxico. En nuestro trabajo hemos usado SpaCy[6], que es una biblioteca de código abierto para el Procesamiento del Lenguaje Natural en Python que soporta más de 34 idiomas, entre ellos el español.

El analizador semántico de SpaCy, según un estudio realizado en 2015 por la universidad Emory[7], es el más rápido y según Medium Corporation tiene un índice de acierto mucho mayor que el analizador sintáctico de NLTK(que era la alternativa a SpaCy que se estudió) [8].Para utilizar el 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.

[5] biblioteca para el acceso de datos, generando clases a partir de las tablas de la base de datos para poder realizar las consultas

[6] https://spacy.io/

[7] https://www.aclweb.org/anthology/P15-1038

[8] 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.

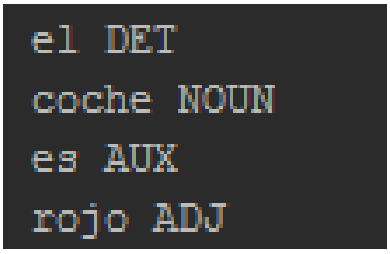

Figura 3.1: Ejemplo de clasificación de palabras

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.

Capítulo 4

Propuesta solución: Aplicación Aprende Fácil

En este capítulo se va a describir la solución propuesta para resolver los problemas mencionados en la sección 1.1. En la sección 4.1 se hablará del proceso de diseño de la interfaz de la aplicación. A continuación, en el apartado 4.2 se comentará como está hecha la aplicación hablando de su arquitectura, la base de datos utilizada y todos los detalles de diseño tanto del Backend como del Frontend. Por último, en el apartado 4.3 se detallará la evaluación del proyecto, realizada con usuario finales y las conclusiones obtenidas.

4.1. Diseño de la Aplicación

Como ya se ha explicado en capítulos anteriores, esta aplicación está creada para personas que tienen alguna discapacidad cognitiva, por lo que la aplicación debe ser sencilla de usar y su uso no debe llevar a confusión ni debe hacer sentir al usuario frustrado. 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 que cubra sus necesidades y que tenga en cuenta sus limitaciones. Por todo ello se ha contado con la ayuda de expertos en el diseño de la aplicación. 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 Afanias[1].

[1] https://afanias.org/que-hacemos/educacion/colegio-estudio-3/

4.1.1. Primera Iteración: Iteración Competitiva

En esta primera iteración cada integrante del grupo ha realizado su propio prototipo de la aplicación. En la realización de estos prototipos, 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 para los usuarios finales era mejor mostrar un solo significado o todos en el caso de las palabras polisémicas, si una definición y un ejemplo les iba a ayudar o no y si añadir los pictogramas les sería o no de utilidad. Los prototipos creados 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 y se incluyen las comparaciones con conceptos relacionados.
- Prototipo 4: Se muestran todos los significados de la palabra y se añaden pictogramas[2], haciendo así que la comprensión del concepto sea mucho mas sencilla. Se han utilizado los pictogramas de ARASAAC[3] ya que son los más usados.

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, nos juntamos para hacer una puesta en común y analizar los prototipos creados. Las principales conclusiones del análisis son las siguientes:

- 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 dirigido a usuariso

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

[3]http://www.arasaac.org/

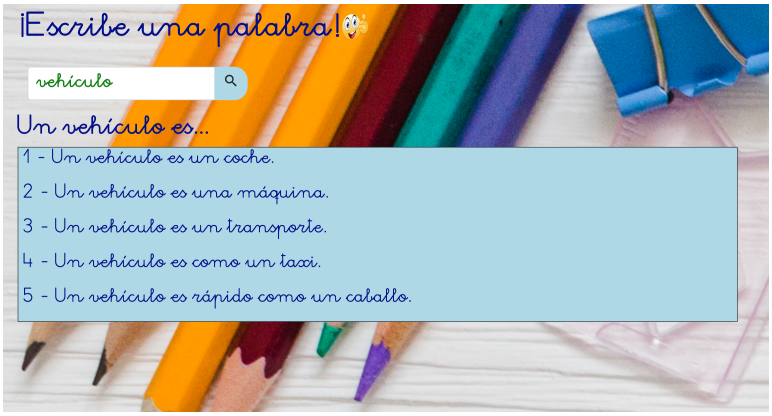

Figura 4.1: Prototipo 1 de Pablo

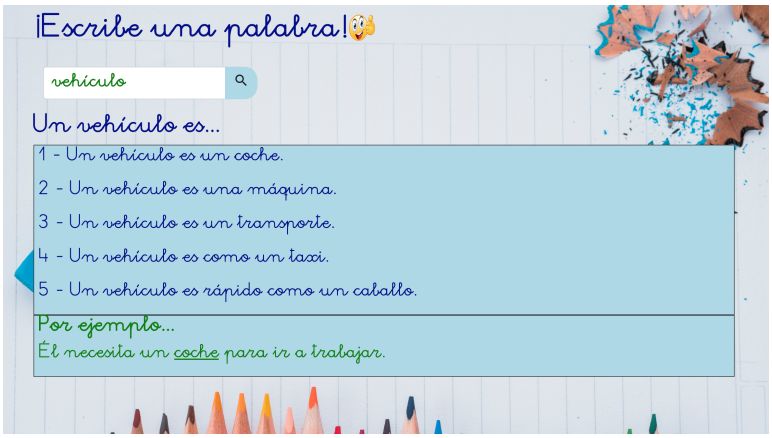

Figura 4.2: Prototipo 2 de Pablo

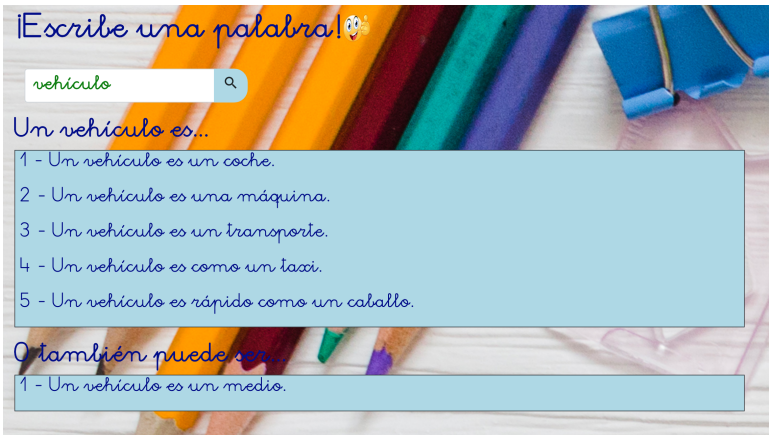

Figura 4.3: Prototipo 3 de Pablo

Figura 4.4: Prototipo 4 de Pablo

Figura 4.5: Prototipo 1 de Irene

de cualquier edad. Por último, los resultados del prototipo de Irene, se muestran en color mostaza indicando de esta forma que se puede pinchar en ellos para realizar 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 como un taxi*) y finalmente las analogías (*Un vehículo es rápido como un caballo*). En los prototipos de Irene se muestra primero la analogía (*Un vehículo es tan rápido como un caballo*), después las metáforas (*Un vehículo es un coche*) y por último los símiles (*Un vehículo es como una ambulancia*).

- El número de resultados mostrados también es distinto en ambos prototipos.

Figura 4.6: Prototipo 2 de Irene

Figura 4.7: Prototipo 3 de Irene

En los prototipos de Irene se muestra únicamente una analogía, una metáfora y un símil para cada significado de la palabra buscada. Y en los prototipos de Pablo se muestran varios resultados para cada tipo.

- En el prototipo 3 de Pablo se 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 se 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

Figura 4.8: Prototipo 4 de Irene

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 asociado a la palabra 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 la apariencia de la interfaz se eligió el de Irene por tener una interfaz más atractiva y dirigida a un colectivo más amplio y no solo a niños como el de Pablo.
- Se eligió la palabra Portero para utilizarla como ejemplo en todos los prototipos, ya que al ser una palabra polisémica se pueden obtener resultados distintos según su contexto y permite realizar comparaciones con conceptos sencillos y fáciles de entender.
- 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”). Este cambio se realizó ya que es preferible ofrecer en un primer momento resultados directos con la palabra buscada y posteriormente ofrecer comparaciones del mismo.
- Se eliminó la palabra *tan* en la plantilla usada para las analogías, ya que no hace falta añadir ningún adverbio de cantidad para mostrar el resultado.
- 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 decidió que era mejor mostrar todos los resultados obtenidos para dicho concepto, para así aumentar la probabilidad de que alguno de los resultados ayudase al usuario a entender el concepto.
- Se decidió que si la palabra solo tenía un significado se eliminaba el número de la ficha, para no llevar a confusión.
- 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, ya que así se ofrece más formas de poder entender el concepto.
- 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, si mantenerlo siempre visible o solo cuando el usuario pulse la 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 las palabras polisémicas?
- ¿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, y de las propiedades que comparten los conceptos relacionados?
- ¿La forma de mostrar los resultados es correcta, o es mejor otra disposición?

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 Afanias[4] 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:

[4] https://afanias.org/que-hacemos/educacion/colegio-estudio-3/

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

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

- Modificar el color amarillo-mostaza de los textos por un color más oscuro que contraste más con el fondo blanco.
- 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.

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

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

- 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 poner el pictograma junto a la palabra puede llevar a confusión a los usuarios finales al pensar que sería otra palabra más para leer.
- Introducción de distintas personalizaciones:

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

- 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 creo el diseño final de la aplicación, que se puede ver en la Figura 4.15.

4.2. Implementación de la Aplicación

En esta sección, se va a hablar de los detalles de implementación de la aplicación. En el apartado 4.2.1 se describirá la arquitectura del proyecto, los distintos elementos que la componen como pueden ser la base de datos, el backend y el frontend En la sección 4.2.2 se detallará el diseño de la base de datos así como las tablas que la componen. En la sección 4.2.3 se explicará la red semántica elegida para implementar la aplicación, junto con las pruebas realizadas para su elección. Por último, en los apartados finales de la presente sección, se hablarán de los detalles de implementación de la aplicación, tanto

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

de la lógica interna en el apartado 4.2.4 donde se describirán los distintos servicios web implementados, como de la vista en el apartado 4.2.5 donde se describirán los elementos que la forman y su funcionalidad.

4.2.1. Arquitectura de Aprende Fácil

En este apartado se describirá la arquitectura de la aplicación.

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

Figura 4.15: Diseño final de la aplicación

los resultados.

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

La aplicación esta desplegada en un contenedor[5] proporcionado por el grupo NiL de la Universidad Complutense de Madrid. Este contenedor no tenía instalado ningún servicio por lo que se tuvo que realizar la instalación de todo lo necesario para Python, pyp, git, mysql, etc. Una vez 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.

4.2.2. 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 siguientes tablas:

- Se ha hecho uso de la base de datos propia de WordNet, llamada MCR (*Multilingual Central Repository*)[6], 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:

[5] holstein.fdi.ucm.es/tfg-analogias
[6] http://adimen.si.ehu.es/web/MCR/

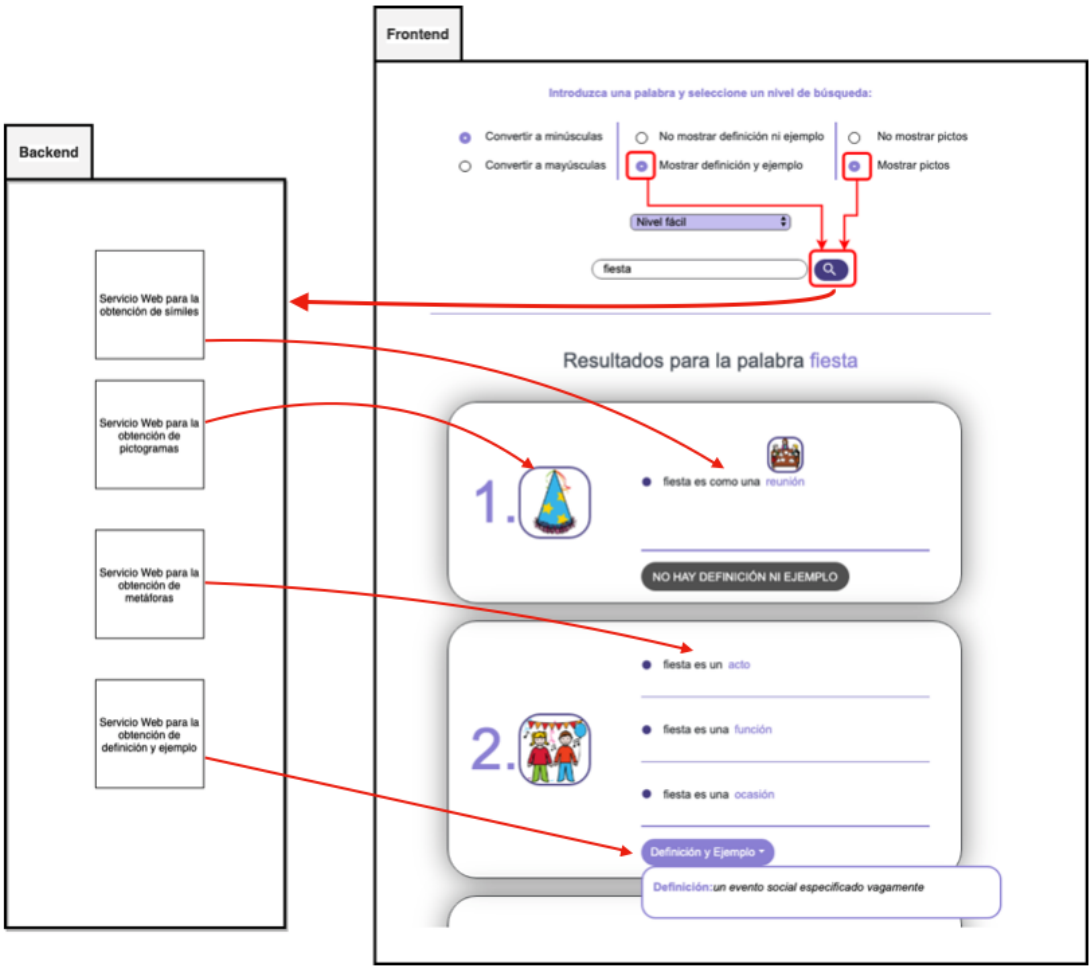

Figura 4.16: Relación entre *Backend* y *Frontend*

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

De la tabla "wei_spa-30_relation", solamente se han utilizado estos atributos:

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

De la tabla "wei_spa-30_examples", se han utilizado los siguientes atributos:

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

De la tabla "wei_spa-30_synset", se han utilizado los siguientes atributos:

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

- 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".

4.2.3. 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 la tarea de encontrar los conceptos relacionados con un concepto dado. De todas las posibilidades presentadas en la sección 2.1 solo hay dos redes semánticas disponibles para el castellano: WordNet y ConceptNet. Para decidir con cual nos quedamos, realizamos una evaluación de ambas redes. En las siguientes secciones detallamos como se llevo a cabo esta evaluación y los resultados obtenidos.

4.2.3.1. 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: tecnológico[7], deportivo[8] y político[9]. (Los artículos se pueden consultar 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 fáciles[10] (análisis cuantitativo). Eligiendo como parámetros: porcentaje de palabras introducidas que generan algún resultado, porcentaje de palabras solo con términos relacionados, porcentaje de palabras solo con sinónimos, porcentaje de palabras con ambos 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 la lista de 10.000 palabras fáciles.El parámetro analizado en este caso era 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.

4.2.3.2. Resultados cuantitativos

En la Tabla 4.17 se muestran los resultados de la prueba cuantitativa realizada, como se puede observar, los resultados son muy parecidos, aunque WordNet encuentra muchas palabras más que ConceptNet.

En el caso de la primera lista utilizada (1.000 palabras fáciles) ConceptNet encontró algún resultado para el 32,8 % de las palabras introducidas mientras que WordNet encontró resultados para el 37,4 %. En el caso de los términos relacionados, el 21,28 % de las palabras encontradas por ConceptNet solo tenían términos relacionados, mientras que en el caso de WordNet fue del 17,3 %. Para el porcentaje de palabras solo con sinónimos los resultados fueron de 10,51 % para ConceptNet y 19,47 % para WordNet. Por último, ni ConceptNet ni WordNet encontraron palabras con términos relacionados y sinónimos.

Para las 5.000 palabras fáciles, las palabras con resultado han sido el 50,4 % para el caso de ConceptNet y 43,5 % en el caso de WordNet. El porcentaje de palabras solo con términos relacionados ha sido de 24,91 % en ConceptNet y 13,56 % en WordNet. En los sinónimos, el 17,24 % de las palabras utilizando ConceptNet tenían solo sinónimos y el 12,98 % con

[7] https://elpais.com/elpais/2019/04/12/ciencia/1555061040_073105.html
[8] https://elpais.com/deportes/2019/04/22/actualidad/1555956020_022201.html
[9] https://elpais.com/politica/2019/04/23/actualidad/1556019953_831941.html
[10] listas de 1.000, 5.000 y 10.000 palabras más utilizadas en castellano según la RAE

		Conceptnet	Wordnet
Lista 1.000 palabras	% palabras con algún resultado	31,8	**37,4**
	% palabras con solo términos relacionados	**21,28**	17,3
	% palabras con solo sinónimos	10,51	**19,47**
	% palabras con ambos	**0**	**0**
Lista 5.000 palabras	% palabras con algún resultado	**50,4**	43,5
	% palabras con solo términos relacionados	**24,91**	13,56
	% palabras con solo sinónimos	**17,24**	12,98
	% palabras con ambos	8,25	**16,96**
Lista 10.000 palabras	% palabras con algún resultado	**56,6**	45
	% palabras con solo términos relacionados	**24,6**	12,78
	% palabras con solo sinónimos	**17,87**	8,35
	% palabras con ambos	14,13	**23,87**
TOTAL PALABRAS GENERADAS		2.652	**14.172**

Figura 4.17: Tabla de resultados de la prueba cuantitativa

WordNet. El porcentaje de palabras con sinónimos y términos relacionados fue de 8,25 % con ConceptNet y 16,96 % con WordNet.

Para las 10.000 palabras fáciles, los resultados han sido: ConceptNet ha encontrado algún resultado para el 56,6 % de frente al 45 % de WordNet. El porcentaje de palabras con solo términos relacionados ha sido del 24,6 % usando ConceptNet frente al 12,78 % de WordNet. En el caso de los sinónimos, ConceptNet ha encontrado resultados para el 17,87 % de las palabras y WordNet para el 8,35 %. Los porcentajes de palabras que tenían tanto sinónimos como términos relacionados han sido con ConceptNet del 14,13 % y con WordNet del 23,84 %. 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 generado por ConceptNet, mientras que WordNet ha generado 14.172.

4.2.3.3. Análisis cualitativo

		ConceptNet	WordNet
Lista 10.000 palabras	% sinónimos correctos	53,03	**60,73**
	% términos relacionados correctos	55,19	**55,3**

Figura 4.18: Tabla de resultados de la prueba cualitativa

El análisis cualitativo consistió en analizar una a una de manera manual, si las palabras obtenidas por ConceptNet y WordNet eran correctas (entendiendo como correctas que tengan relación directa con la palabra buscada inicialmente).Como se puede observar, en la Tabla 4.18 los resultados son muy parejos para ambas redes semánticas. El análisis se ha realizado únicamente con las 10.000 palabras fáciles ya que esta lista contiene a las otras dos. Los resultados 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).

4.2.3.4. 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 aunque ConceptNet deja menos palabras sin ningún resultado, WordNet obtiene más palabras con ambos, por lo que se podrán generar más símiles y metáforas. 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, con la que se han obtenido mejores resultados ha sido WordNet ya que ha obtenido porcentajes más altos tanto para los sinónimos como para los términos relacionados.

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

4.2.4. Backend de Aprende Fácil: Servicios Web de la aplicación

En este capítulo se explicará la implementación de los distintos servicios web. En los apartados 4.2.4.1, 4.2.4.2 y 4.2.4.3 se describirán los servicios web que se utilizan para la obtención de sinónimos, hiperónimos e hipónimos fáciles respectivamente. A continuación, se hablará en los puntos 4.2.4.4 y 4.2.4.5 sobre los servicios web utilizados para crear las metáforas y los símiles que finalmente leerá el usuario. Por último, se describirá el servicio web para la obtención de definiciones y ejemplos en la sección 4.2.4.6 y en la sección 4.2.4.7 se hablará de los servicios web para la obtención de pictogramas.

4.2.4.1. Servicio Web para la Obtención de Sinónimos Fáciles

Servicio web[11] que recibe como entrada una palabra y un nivel de búsqueda, y devuelve todos los sinónimos fáciles en formato JSON. La *palabra* será el

[11] https://holstein.fdi.ucm.es/tfg-analogias/easysynonym/json/word=*palabra*&level=*nivel*

concepto a buscar y *nivel* será el nivel 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 de WordNet. A través de una *queryset* sobre la tabla *WeiSpa30Variant*, se obtienen todos los *offsets* cuya palabra sea igual a la introducida. Si se obtienen resultados, se vuelve a buscar en la base de datos de WordNet, y se realiza de nuevo una *queryset* sobre la misma tabla nombrada con anterioridad, para obtener las palabras asociadas a 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. Si la palabra se encuentra en las palabras más usadas de la RAE, se añade a su correspondiente campo del JSON. En la Figura 4.19 se puede ver el Diagrama de Flujo del servicio web.

Si no se obtiene ningún resultado se devolverá el JSON solamente con un campo que guarda la palabra que se ha introducido, y si hay resultados devolverá el JSON con el campo que guarda la palabra buscada, y un objeto donde se guarda el *offset* de dicho concepto y un array de los sinónimos fáciles.

Por ejemplo, si se realiza la búsqueda[12] para la palabra "tren" con un nivel de búsqueda 3 (nivel avanzado), se devuelve el JSON que se puede ver en la Figura 4.20.

En primer lugar aparece la palabra buscada, en este caso "tren" y a continuación un objeto que incluye tanto el *offset* de la palabra como la lista de sinónimos fáciles .

4.2.4.2. Servicio Web para la Obtención de Hiperónimos Fáciles

Servicio web[13] que dada una palabra y un nivel de búsqueda, devuelve todos los hiperónimos fáciles de dicha palabra en formato JSON. La *palabra* será el concepto a buscar y *nivel* será el nivel de búsqueda, igual que en el caso explicado anteriormente en el servicio web para la obtención de sinónimos.

[12] https://holstein.fdi.ucm.es/tfg-analogias/easysynonym/json/word=tren&level=3
[13] https://holstein.fdi.ucm.es/tfg-analogias/easyhyperonym/json/word=*palabra*&level=*nivel*

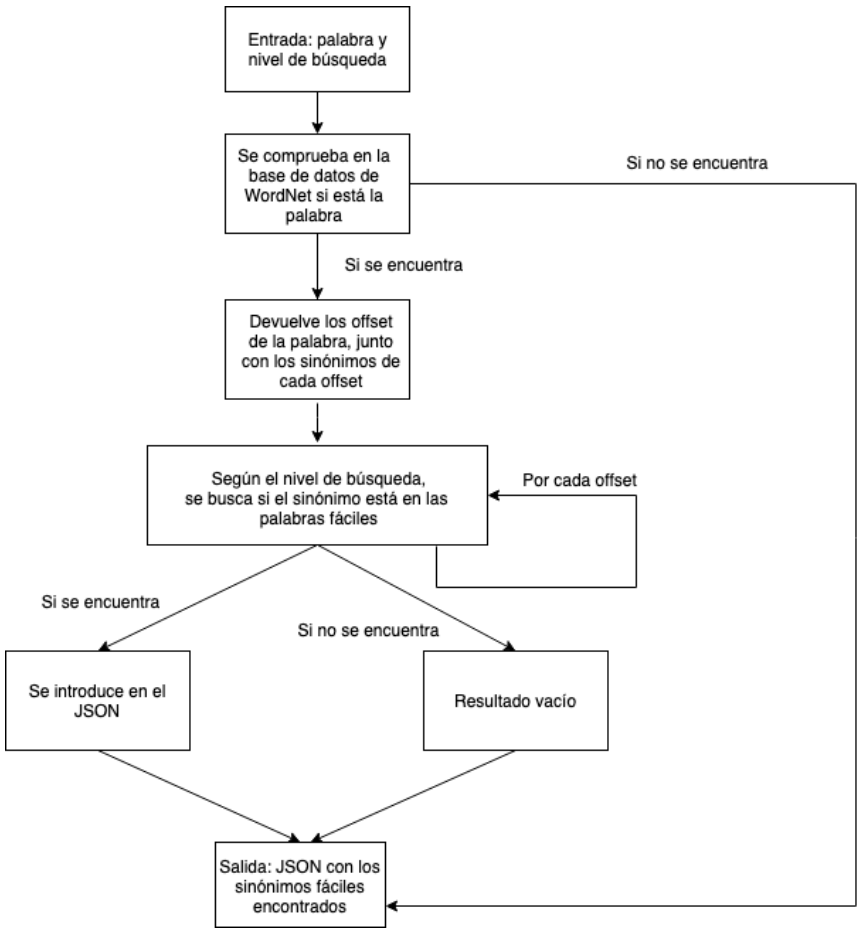

Figura 4.19: Diagrama de flujo del Servicio Web para la obtención de Sinónimos Fáciles

Una vez introducida la palabra y el nivel, se realiza una consulta a la base de datos de WordNet. A través de una *queryset* sobre la tabla *WeiSpa30Variant*, se obtienen todos los *offsets* de la palabra. Por cada *offset* volveremos a buscar en la tabla *WeiSpa30Relation* de la base de datos de WordNet para obtener los *offsets* que se encuentran en la columna *sourceSynset*, es decir, se obtienen los hiperónimos. Después, con cada *offset* obtenido de esta *queryset*, se realizará la búsqueda en la tabla *WeiSpa30Variant* para obtener las palabras cuyo identificador sea igual que el *offset* de *sourcetSynset*. Cada palabra asociada a los *offsets* se buscará en una de las tres tablas de la RAE (en función del nivel introducido) y se buscará si alguna de estas palabras se encuentra en dicha tabla. Si el resultado es positivo se añadirá al JSON. En la Figura 4.21 se puede ver el Diagrama de Flujo del servicio web.

```
[
  {
    "word": "tren"
  },
  {
    "offset": "spa-30-04468005-n",
    "synonyms": [
      "ferrocarril",
      "trenes"
    ]
  },
  {
    "offset": "spa-30-02959942-n",
    "synonyms": [
      "coche"
    ]
  },
  {
    "offset": "spa-30-03431745-n",
    "synonyms": [
      "cambio",
      "marcha"
    ]
  }
]
```


Figura 4.20: JSON para la palabra "tren" devuelto por el servicio web para la obtención de sinónimos fáciles

Por ejemplo, si se realiza la búsqueda[14] para la palabra "nivel" con un nivel de búsqueda 1 (nivel fácil), se obtendría el JSON mostrado en la Figura 4.22.

En este JSON, en primer lugar aparece la palabra buscada, en este caso "nivel" y a continuación un objeto que incluye el *offset* del hiperónimo fácil, el *offsetFather* (*offset* del *synset* "nivel" con el que están relacionados los hiperónimos) y la lista de hiperónimos fáciles.

4.2.4.3. Servicio Web para la Obtención de Hipónimos fáciles

Servicio web[15] que dada una palabra y un nivel de búsqueda, devuelve todos los hipónimos fáciles de dicha palabra en formato JSON. La *palabra* será el concepto a buscar y *nivel* será el nivel de búsqueda.

La implementación de este servicio web es igual que el servicio web para la obtención de hiperónimos fáciles, con la única diferencia de que en este en vez de obtener los *offset* de la columna *sourceSynset*, se obtiene de *targetSynset*, es decir, los hipónimos. En la Figura 4.23 se puede ver el Diagrama de Flujo del servicio web.

Por ejemplo, si se realiza la búsqueda[16] para la palabra "tren" con un nivel de búsqueda 3 (nivel avanzado), se obtiene el JSON mostrado en la Figura 4.24. En este JSON, en primer lugar aparece la palabra buscada, en

[14] https://holstein.fdi.ucm.es/tfg-analogias/easyhyperonym/json/word=nivel&level=1
[15] https://holstein.fdi.ucm.es/tfg-analogias/easyhyponym/json/word=*palabra*&level=*nivel*
[16] https://holstein.fdi.ucm.es/tfg-analogias/easyhyponym/json/word=tren&level=3

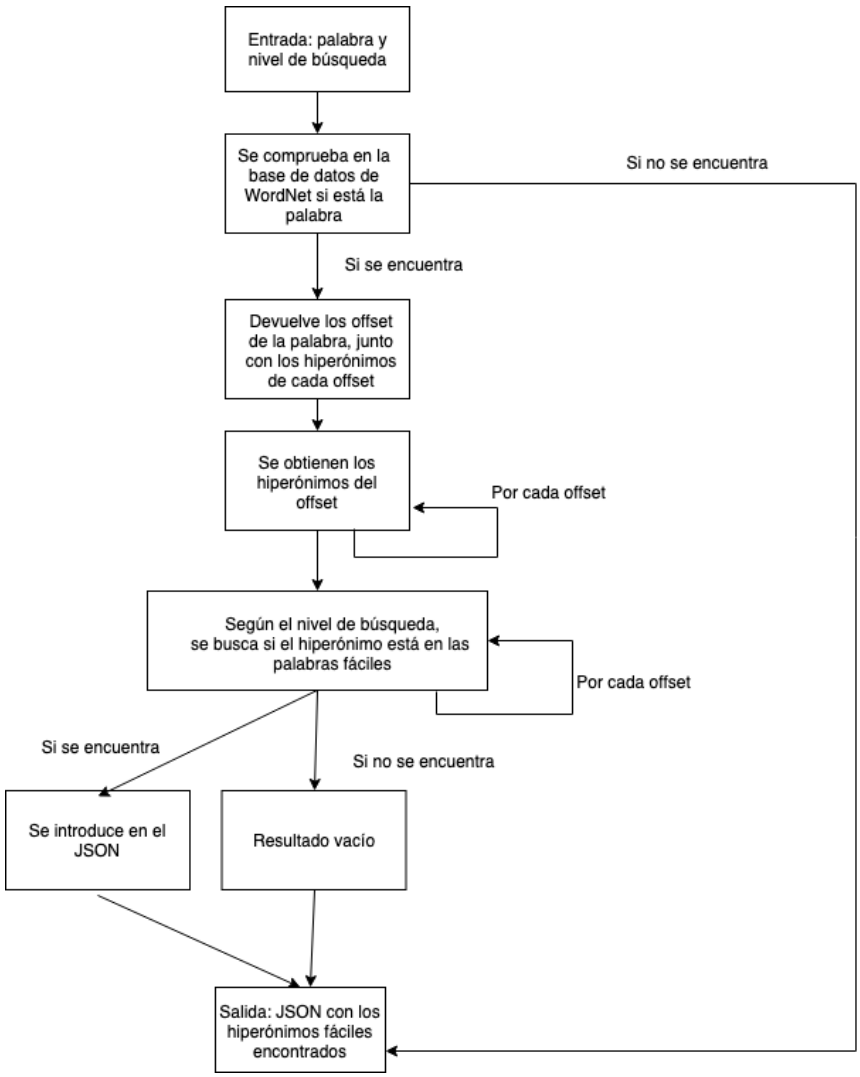

Figura 4.21: Diagrama de flujo del Servicio Web para la obtención de Hiperónimos Fáciles

este caso "tren" y a continuación un objeto que incluye el *offset* del hipónimo fácil, el *offsetFather* (*offset* del *synset* "tren" con el que están relacionados los hipónimos) y la lista de hipónimos fáciles.

En primer lugar aparece la palabra buscada, en este caso "tren" y a continuación un objeto que incluye el *offset* del hipónimo fácil , la lista de hipónimos fáciles así como el *offsetFather*, este identificador corresponde al *offset* de uno de los *synsets* de inmueble.

```
[
  {
    "word": "nivel"
  },
  {
    "offset": "spa-30-00024720-n",
    "offsetFather": "spa-30-13939892-n",
    "hyperonyms": [
      "estado"
    ]
  },
  {
    "offset": "spa-30-13945919-n",
    "offsetFather": "spa-30-14429985-n",
    "hyperonyms": [
      "estado",
      "lugar",
      "posición"
    ]
  }
]
```


Figura 4.22: JSON para la palabra "nivel" devuelto por el servicio web para la obtención de hiperónimos fáciles

4.2.4.4. Servicio Web para la Obtención de Metáforas

Servicio web[17] que dada una palabra y un nivel de búsqueda, devuelve todos las metáforas para dicha palabra en formato JSON. La *palabra* será el concepto a buscar y *nivel* será el nivel de búsqueda. Las metáforas devueltas estarán formadas por los sinónimos e hiperónimos de la palabra de entrada.

Una vez introducida la palabra, se realiza una consulta a la base de datos de WordNet. A través de una *queryset* a la tabla *WeiSpa30Variant*, se obtienen todos los *offsets* de la palabra. A continuación, se invoca al servicio web para la obtención de sinónimos y al servicio web para la obtención de hiperónimos, obteniendo así los sinónimos e hiperónimos fáciles. Después, se invoca a otro servicio web[18] (facilitado por el grupo NiL[19] de la Facultad de Informática) que permite obtener un JSON con la información morfológica de la palabra introducida, en este caso se introduce cada sinónimo e hiperónimos fácil. Con esta información, se construye cada metáfora correctamente y se añaden al JSON.

En la Figura 4.26 se puede ver el Diagrama de Flujo del servicio web.

Por ejemplo, si se realiza la búsqueda[20] para la palabra "sala" con un nivel de búsqueda 1 (nivel fácil), se obtendría el JSON mostrado en la Figura

[17] https://holstein.fdi.ucm.es/tfg-analogias/metaphor/json/word=*palabra*&level=*nivel*
[18] https://holstein.fdi.ucm.es/nlp-api/analisis/*palabra*
[19] http://nil.fdi.ucm.es
[20] https://holstein.fdi.ucm.es/tfg-analogias/metaphor/json/word=sala&level=1

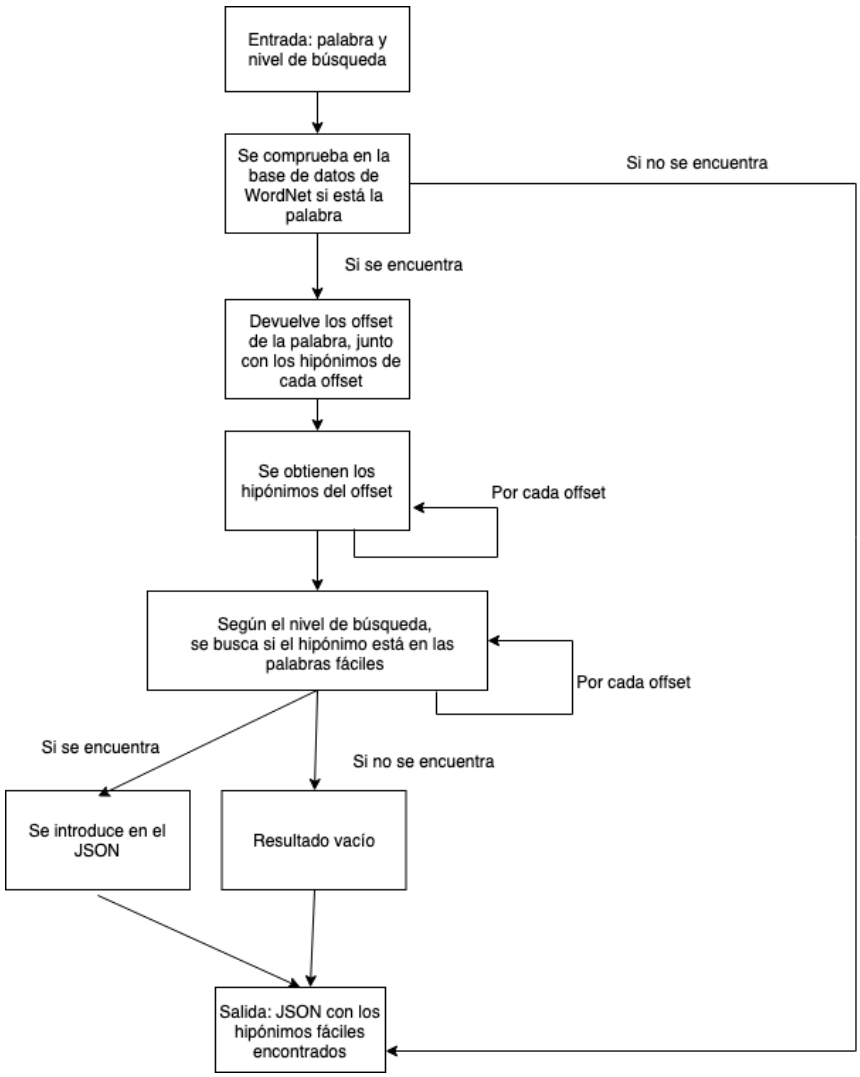

Figura 4.23: Diagrama de flujo del Servicio Web para la obtención de Hipónimos Fáciles

4.25. En primer lugar aparece la palabra buscada, en este caso "sala" y a continuación un objeto que incluye el *offset* del sinónimo o hiperónimo fácil, en el caso de que sea un hiperónimo aparecerá el *offsetFather*, ya explicado en apartados anteriores. A continuación, se muestra la lista de metáforas (es un cuarto, es una zona) y por último un campo que indica si es un sinónimo o un hiperónimo.

```
[
  {
    "word": "tren"
  },
  {
    "offset": "spa-30-80000592-n",
    "offsetFather": "spa-30-04468005-n",
    "hyponyms": [
      "metro"
    ]
  },
  {
    "offset": "spa-30-04520480-n",
    "offsetFather": "spa-30-02959942-n",
    "hyponyms": [
      "cabina",
      "coche",
      "van"
    ]
  }
]
```


Figura 4.24: JSON para la palabra "tren" devuelto por el servicio web para la obtención de hipónimos fáciles


```
[
  {
    "word": "sala"
  },
  {
    "offset": "spa-30-04105893-n",
    "metaphor": [
      "es un cuarto"
    ],
    "type": "SYNONYM"
  },
  {
    "offset": "spa-30-02735688-n",
    "offsetFather": "spa-30-04105893-n",
    "metaphor": [
      "es una zona",
      "es una área"
    ],
    "type": "HYPERONYM"
  }
]
```


Figura 4.25: JSON para la palabra "sala" devuelto por el servicio web para la obtención de metáforas

4.2.4.5. Servicio Web para la Obtención de Símiles

Servicio web[21] que dada una palabra y un nivel de búsqueda, devuelve todos los símiles para dicha palabra en formato JSON. La *palabra* será el

[21] https://holstein.fdi.ucm.es/tfg-analogias/simil/json/word=*palabra*&level=*nivel*

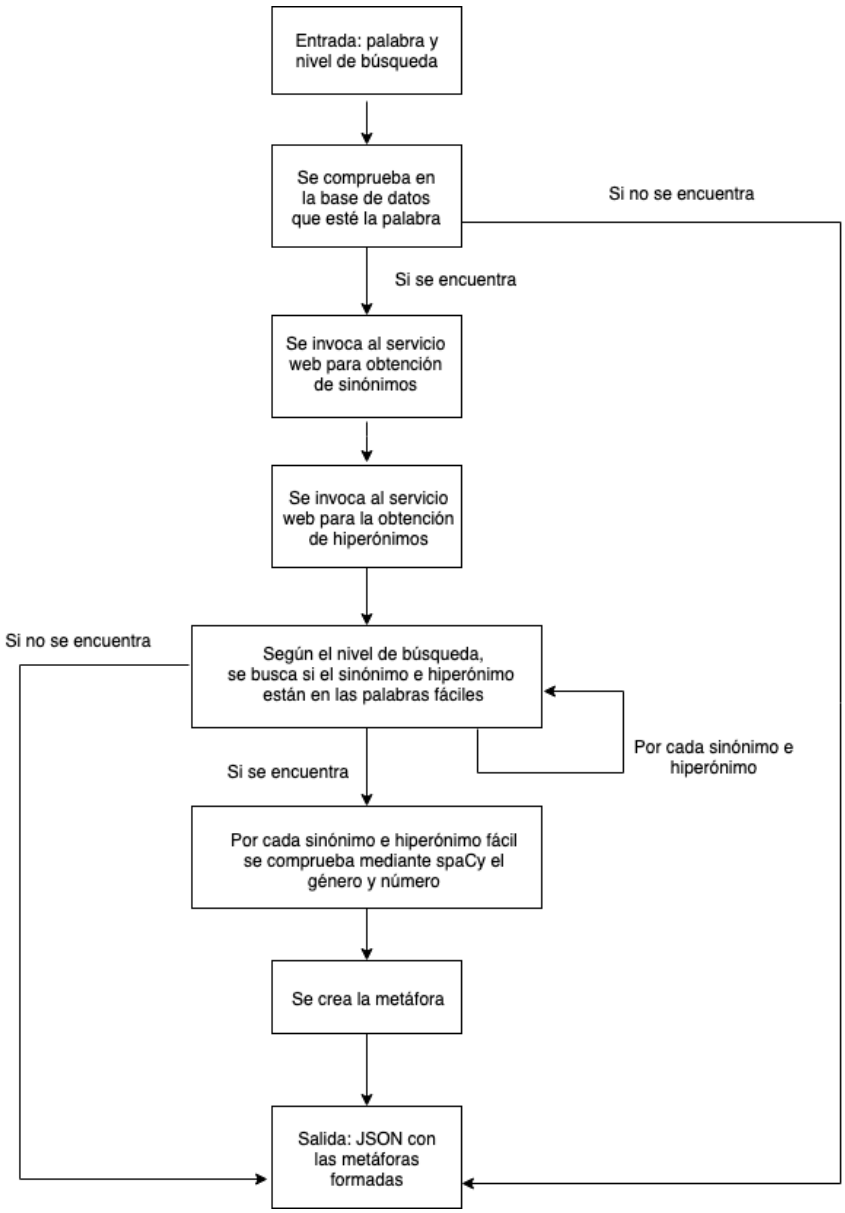

Figura 4.26: Diagrama de flujo del Servicio Web para la obtención de Metáforas

concepto a buscar y *nivel* será el nivel de búsqueda. Los símiles devueltos estarán formados por los hipónimos de la palabra de entrada.

La implementación de dicho servicio web es igual que la realizada en el servicio web para la obtención de metáforas pero con la diferencia de que este invoca únicamente al servicio web para la obtención de hipónimos.

En la Figura 4.28 se puede ver el Diagrama de Flujo del servicio web.

```
[
  {
    "word": "palabra"
  },
  {
    "offset": "spa-30-06752695-n",
    "offsetFather": "spa-30-06286395-n",
    "simil": [
      "es como una forma",
      "es como un lenguaje"
    ]
  },
  {
    "offset": "spa-30-07148192-n",
    "offsetFather": "spa-30-07140659-n",
    "simil": [
      "es como un congreso"
    ]
  }
]
```


Figura 4.27: JSON para la palabra "palabra" devuelto por el servicio web para la obtención de símiles

Por ejemplo, si se realiza la búsqueda[22] para el concepto "palabra" con un nivel de búsqueda 1 (nivel fácil), se obtendría el JSON mostrado en la Figura 4.27. En primer lugar aparece la palabra buscada, en este caso "palabra" y a continuación un objeto que incluye el *offset* del hipónimo fácil, el *offsetFather*, ya explicado en apartados anteriores. Y por último, una lista de símiles (es como una forma, es como un lenguaje).

4.2.4.6. Servicio Web para la Obtención de Definiciones y Ejemplos

Servicio web[23] que dada una palabra y un nivel de búsqueda, devuelve todos las definiciones y ejemplos para dicha palabra en formato JSON. La *palabra* será el concepto a buscar y *nivel* será el nivel de búsqueda.

La implementación de dicho servicio web se basa en que introduciendo una palabra y un nivel de búsqueda, este invoca al servicio web para la obtención de metáforas y al servicio web para la obtención de símiles. Por cada metáfora y por cada símil se realiza una consulta a la base de datos mediante una *queryset* a las tablas *WeiSpa30Synset* y *WeiSpa30Examples* respectivamente para obtener las definiciones y los ejemplos.

En la Figura 4.29 se puede ver el Diagrama de Flujo del servicio web.

Por ejemplo, si se realiza la búsqueda[24] para la palabra "agua" con un

[22] https://holstein.fdi.ucm.es/tfg-analogias/simil/json/word=palabra&level=1
[23] https://holstein.fdi.ucm.es/tfg-analogias/def_examen/json/word=*palabra*&level=*nivel*
[24] https://holstein.fdi.ucm.es/tfg-analogias/metaphor/json/word=agua&level=1

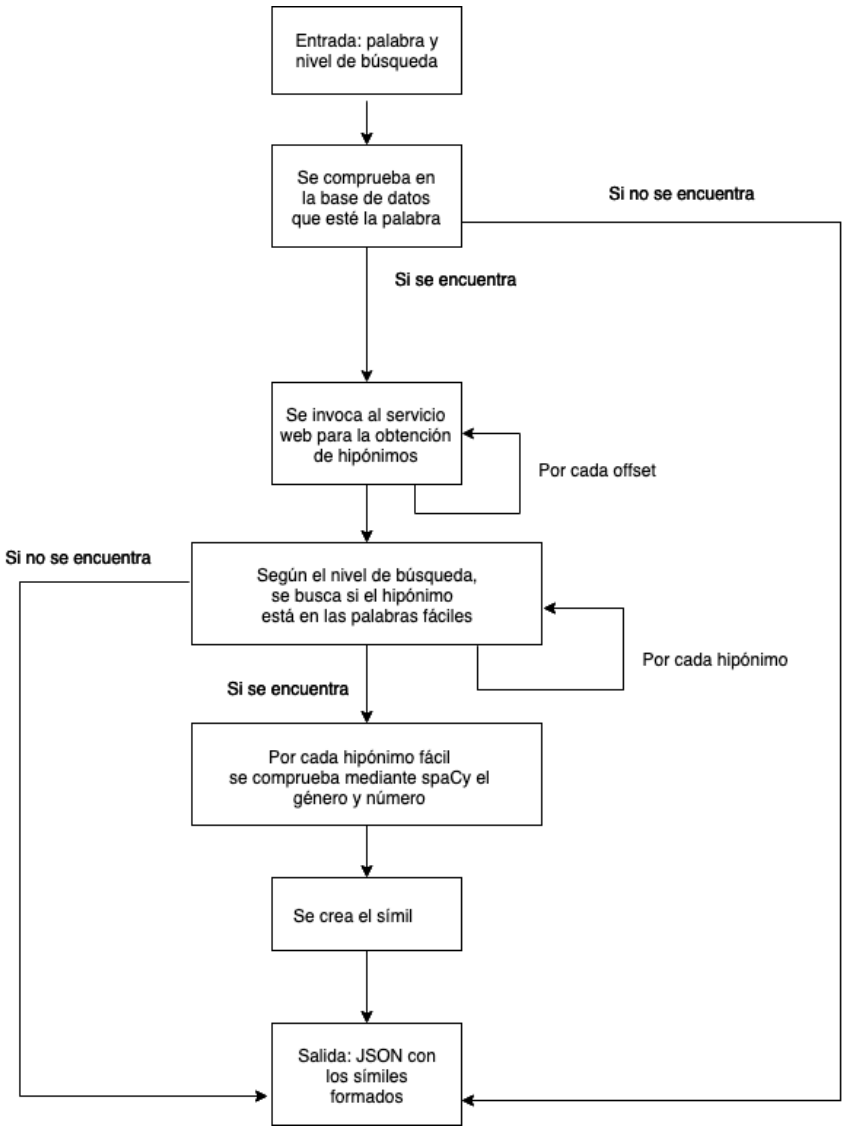

Figura 4.28: Diagrama de flujo del Servicio Web para la obtención de Símiles

nivel de búsqueda 1 (nivel fácil), se obtendría el JSON mostrado en la Figura 4.30. En primer lugar aparece la palabra buscada, en este caso "agua" y a continuación un objeto metáforas que incluye un campo *example*, el *offset*, la definicion, el *offsetFather* y por último, el tipo de palabra que es (sinónimo, hiperónimo o hipónimo).

4.2.4.7. Servicios Web para la Obtención de Pictogramas

Se han implementado dos servicios web para la obtención de pictogramas: uno que recibiendo una palabra se obtiene el pictograma correspondiente y

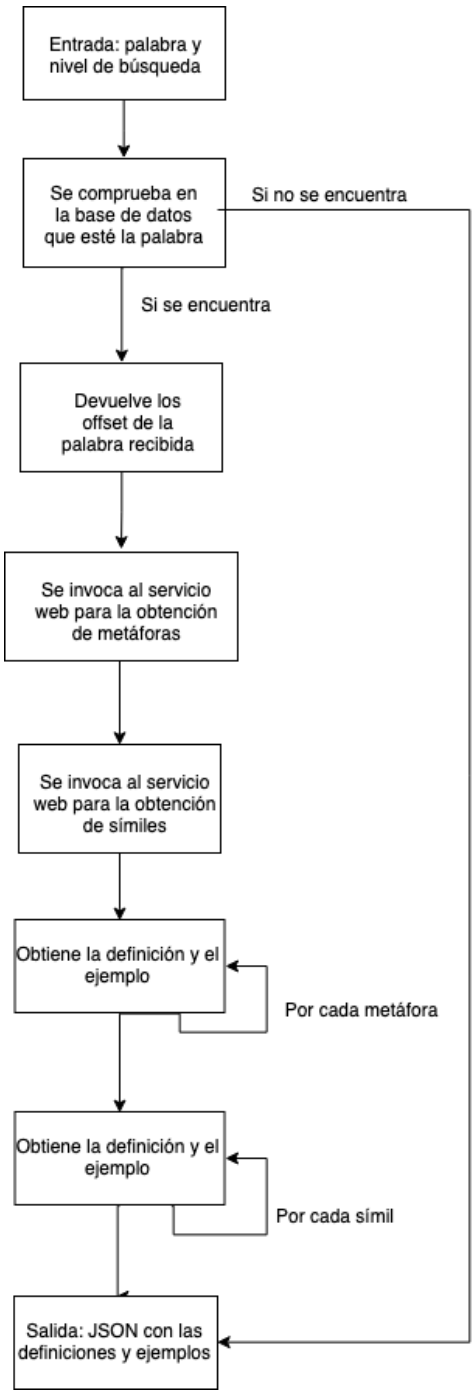

Figura 4.29: Diagrama de flujo del Servicio Web para la obtención de definiciones y ejemplos

```
[
  {
    "word": "agua"
  },
  {
    "metaphor": [
      {
        "example": [],
        "offset": "spa-30-00002452-n",
        "definition": "una entidad separada y autocontenida",
        "offsetFather": "spa-30-09225146-n",
        "type": "HYPERONYM"
      },
      {
        "example": [],
        "offset": "spa-30-03315023-n",
        "definition": "un edificio o un lugar que ofrece un servicio
          determinado o se utiliza para una industria concreta",
        "offsetFather": "spa-30-04562658-n",
        "type": "HYPERONYM"
      }
    ]
  },
  {
    "simil": [
      {
        "example": [],
        "offset": "spa-30-80000216-n",
        "definition": "",
        "offsetFather": "spa-30-09225146-n",
        "type": "HYPONYM"
      }
    ]
  }
]
```


Figura 4.30: JSON para la palabra "agua" devuelto por el servicio web para la obtención de definiciones y ejemplos

otro que recibe un *offset* en vez de la palabra para devolver el pictograma.

Para acceder al primero se debe de realizar una petición GET[25].

En la Figura 4.31 se puede ver el Diagrama de Flujo del servicio web.

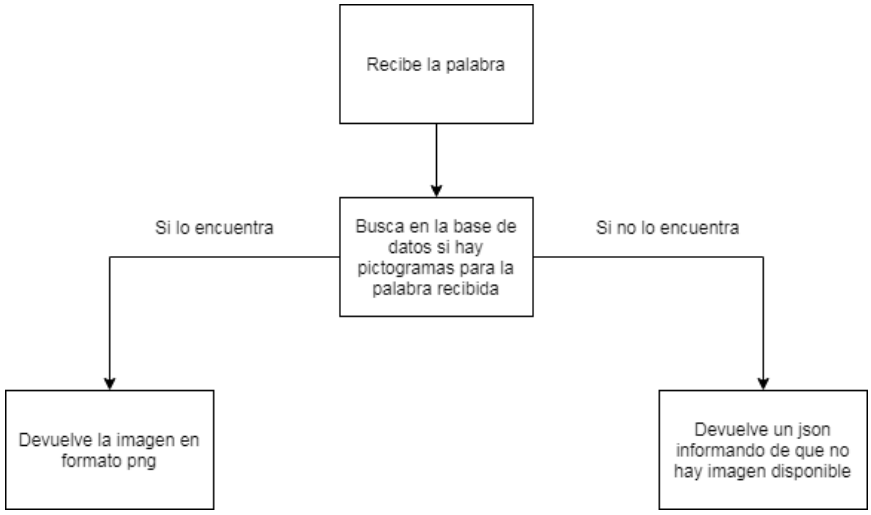

Figura 4.31: Diagrama de flujo del Servicio Web para la obtención de pictogramas introduciendo una palabra

[25] https://holstein.fdi.ucm.es/tfg-analogias/image_word/*palabra*

El servicio web, recibe la palabra por parámetro y busca en la tabla pictos de la base de datos si hay alguna palabra que coincida, si es así, recupera la imagen, la decodifica y la muestra en formato PNG.

Por ejemplo, si se quisiera obtener el pictograma de la palabra coche se haría de esta manera:

https://holstein.fdi.ucm.es/tfg-analogias/image_word/coche

Por otro lado, para acceder al segundo servicio web para la obtención de pictogramas se debe hacer una petición GET[26]

En la Figura 4.32 se puede ver el Diagrama de Flujo de este segundo servicio web, como se puede observar, es similar al anterior, cambiando solo el parámetro palabra por *offset*.

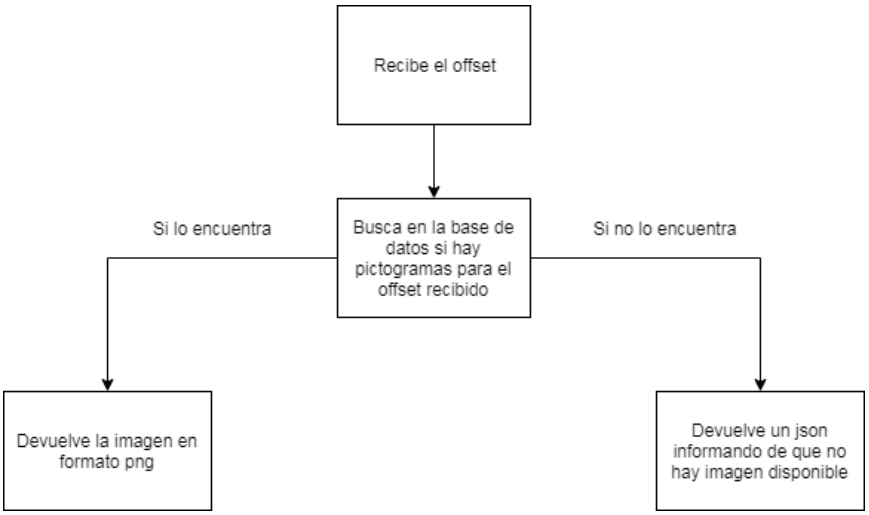

Figura 4.32: Diagrama de flujo del Servicio Web para la obtención de pictogramas introduciendo un offset

4.2.5. 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.

[26] https://holstein.fdi.ucm.es/tfg-analogias/image_offset/*offset*

En la Figura 4.33 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:

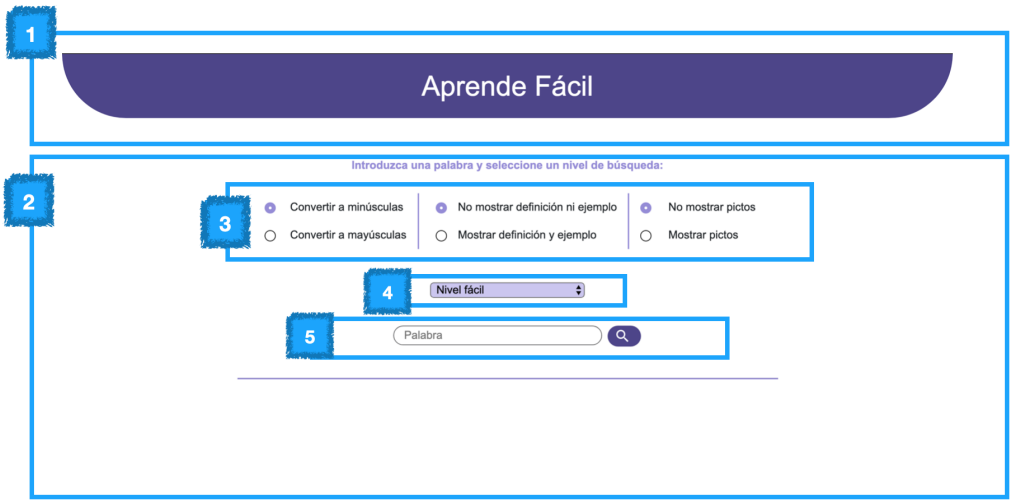

Figura 4.33: Interfaz de Aprende Fácil sin realizar ninguna búsqueda

- 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 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 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 que 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 que debe introducir. Y por otro lado, se encuentra el botón con un icono de una lupa pero que al pasar el ratón por encima tiene un efecto y el botón se alarga, añadiendo a este icono la palabra Buscar. Tanto el input como el botón están dentro de un 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 4.34 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 dinamicamente según el número de resultados que se obtengan y es un 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*

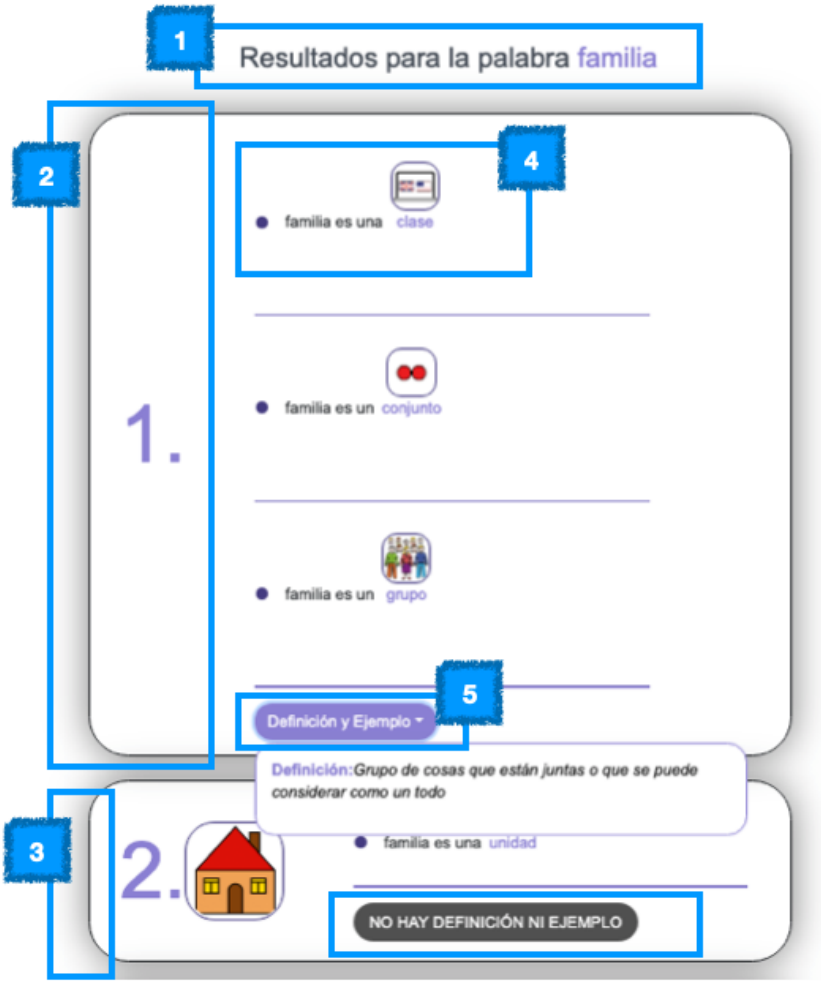

Figura 4.34: Interfaz de Aprende Fácil mostrando resultados

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 desaparecían 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 el, 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.

4.3. Evaluación de Aprende Fácil

La evaluación final de la aplicación, se realizó en el colegio Estudio3 Afanias[27] situado en la Comunidad de Madrid. Este colegio era el mismo donde se realizó la evaluación con los expertos explicado en la sección 4.1.2. Gracias a realizar esta evaluación se pudo probar la funcionalidad de la aplicación final, y de esta forma comprobar si realmente ayudaba a quienes son los usuarios finales o si por lo contrario, la aplicación no les era útil y les ocasionaba mayor confusión al intentar entender el significado de un concepto. En las siguientes secciones se explicará con mayor detalle como se ha realizado dicha evaluación, los resultados obtenidos tras realizarla y el análisis que se ha realizado de los resultados.

4.3.1. Diseño de la evaluación

Se creó un formulario de Google Forms, el cual se puede encontrar en nuestro repositorio de GitHub, dividido en dos secciones: una primera sección donde la primera parte era guiada y el usuario debía introducir distintos conceptos ya predefinidos por nosotros e ir probando diferentes configuraciones de la aplicación. De esta forma se intentó abarcar todas las posibilidades que la aplicación ofrece y obtener así, de primera mano, la opinión de los usuarios. Dentro de la primera sección, existía una segunda parte de uso libre. El usuario podía introducir los conceptos deseados y utilizar la aplicación con

[27] https://afanias.org/que-hacemos/educacion/colegio-estudio-3/

las configuraciones que estimase más oportuno. Por otro lado, existía una segunda sección en el formulario donde se realizaba un cuestionario sobre la usabilidad y las opiniones referentes a la aplicación en general.

En la primera parte del formulario tomábamos datos demográficos: edad y sexo del usuario.

A continuación, se pedía introducir la palabra "puente" con un nivel de búsqueda fácil. El objetivo era comprobar que los usuarios entendían el significado del concepto buscado mediante los resultados mostrados.

En la segunda parte, se pedía introducir la palabra "puente" pero esta vez con nivel de búsqueda medio. La aplicación mostraba más resultados que con el nivel de búsqueda facil, y por ello lo que se quería comprobar era si la búsqueda en este nivel ayudaba a los usuarios a comprender aún más el significado del concepto introducido, o sin embargo, les llevaba a confusión.

En la tercera parte, se pedía al usuario introducir la palabra "puente" con nivel de búsqueda avanzado. Igual que ocurre con el nivel de búsqueda medio, se muestran más resultados para la palabra introducida, de esta forma se puede saber si a los usuarios les ha servido más la búsqueda en un nivel que en otro, o entienden los resultados mostrados en todos los niveles de búsqueda.

En la cuarta parte, se probó a buscar un concepto que devolvía únicamente un conjunto de resultados relacionados entre sí, es decir, una única ficha. La palabra introducida era "dictador" y con un nivel de búsqueda avanzado.

A continuación, en la quinta y sexta parte se introducía la palabra "defensa" con un nivel de búsqueda fácil y "danza" con un nivel de búsqueda avanzado. El objetivo de esta quinta y sexta parte era obtener más de una ficha con resultados. Esto quiere decir que la palabra introducida dispone de distintos significados según el contexto en que se utilice, por lo que el fin principal era comprobar si los usuarios entendían todos ellos.

En la séptima parte, se pedía introducir la palabra "portero" con un nivel de búsqueda avanzado. En esta parte el objetivo era obtener únicamente metáforas con un concepto polisémico y así comprobar si los usuarios entendían correctamente los diferentes resultados obtenidos.

Después, en la octava parte se pedía introducir la palabra "funcionario" con un nivel de búsqueda fácil. En esta parte solamente se quería obtener símiles, e igual que ocurre con las anteriores pruebas el objetivo era comprobar si los usuarios entendían correctamente los resultados obtenidos.

En la novena y décima parte, se pedía introducir la palabra "régimen" con un nivel de búsqueda fácil y "vehículo" con un nivel de búsqueda avanzado. El objetivo principal era obtener resultados que tuvieran tanto metáforas como símiles y comprobar si los resultados obtenidos eran correctos y entendibles para los usuarios o si, por el contrario, les resultaba más útiles unos que otros, o si ningún resultado le facilitaba a entender el significado del concepto introducido.

En la undécima y duodécima parte, el objetivo principal era obtener resultados pero añadiendo la opción de mostrar pictogramas. La palabra a introducir era "detective" y "medalla" , ambos con un nivel de búsqueda fácil. Con esta parte, se quería comprobar si añadiendo la opción de mostrar pictogramas a los usuarios les era más fácil comprender los resultados obtenidos o no. Una vez que se muestran los resultados, se pedía al usuario seleccionar la opción de no mostrar pictogramas y comprobar así que la funcionalidad de mostrar y ocultar los pictogramas se realizaba correctamente.

En la décimotercera parte, se pedía introducir la palabra "familia" seleccionado la opción de mostrar definición y ejemplo. El objetivo principal era comprobar si añadiendo en la búsqueda de un concepto esta opción, les ayudaba aún más a los usuarios a comprender mejor el significado de la palabra. Una vez que se muestran los resultados, se pedía al usuario seleccionar la opción de no mostrar definición y ejemplo, y comprobar así que la funcionalidad de mostrar y ocultar la definición y el ejemplo se realizaba correctamente.

En la décimocuarta y décimoquinta parte, se pedía al usuario seleccionar la opción de mostrar en mayúsculas con la búsqueda realizada en la décimotercera parte y posteriormente seleccionar la opción de mostrar en minúscula. El objetivo principal era comprobar si la aplicación convertía todo el texto de la página a mayúsculas y minúsculas correctamente y saber si para los usuarios era más cómodo leer el texto con una opción u otra.

En la décimosexta parte, se pedía al usuario introducir la palabra "casa" con un nivel de búsqueda fácil y una vez obtenidos los resultados pinchar sobre la palabra "escuela". El objetivo principal era comprobar si la búsqueda se realizaba correctamente sobre el resultado y saber si para los usuarios era útil dicha opción.

Por último, en la décimoséptima parte le pedíamos al usuario que probara la aplicación libremente, introduciendo las palabras que el deseara y seleccionado las distintas opciones que la aplicación ofrece. El objetivo de esta parte, era comprobar si el usuario podía utilizar la aplicación sin necesidad de una guía y si los resultados obtenidos eran correctos y útiles para el usuario.

En la segunda sección del formulario, se realizaba una serie de afirmaciones al usuario relacionadas con la usabilidad de la aplicación, donde ellos mediante una escala *likert* de cinco puntos, siendo 1 totalmente en desacuerdo y 5 totalmente de acuerdo, debían valorar. Las afirmaciones realizadas eran las siguientes:

- La aplicación es fácil de usar.
- Me gustaría usar la aplicación en mi día a día.
- Recomendaría esta aplicación a otras personas.
- Las definiciones se ajustan a los resultados obtenidos por la aplicación.

- Me parece útil poder modificar la opción de mostrar y ocultar la definición y ejemplo con los resultados ya mostrados.
- Los pictogramas se ajustan a la palabra buscada.
- Los pictogramas se ajustan a los resultados obtenidos por la aplicación.
- Me parece útil poder modificar la opción de mostrar y ocultar los pictogramas con los resultados ya mostrados.
- Me parece útil que se pueda hacer la búsqueda pinchando sobre cualquier resultado.
- Me parece útil que las opciones se encuentren siempre visibles arriba.
- Me parece útil que en cada ficha se muestren primero las metáforas y después los símiles.

Y por último, se añadió un apartado de observaciones donde el usuario puede escribir su opinión escrita sobre la aplicación.

Una vez se realizaba el formulario, se enviaba y las respuestas quedaban registradas para su posterior análisis.

4.3.2. Resultados de la evaluación

El día 17 de Mayo de 2019 a las 10:30h tuvo lugar la evaluación final . A esta evaluación acudieron los integrantes del equipo junto con los directores. La evaluación se realizó con dos grupos: 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 tenían un mayor grado de discapacidad.

La idea principal era que cada usuario probará la aplicación y realizara el formulario de forma individual, pero una vez allí nos dimos cuenta que los alumnos presentes según estaba realizado el formulario, les iba a resultar complicado o incluso imposible poder hacerlo. Si en vez de utilizar la escala *likert* numerada, hubiéramos añadido un código de colores donde rojo significa que los resultados obtenidos no eran correctos, naranja para indicar que los resultados son más o menos entendibles, y verde para indicar que todos los resultados son correctos hubiera sido mucho más fácil para ellos, ya que están familiarizados con ese tipo de código de colores en vez de un tipo numérico. Por ello, 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.

Cabe destacar que en ambos grupos no se pudo llegar a realizar el formulario completo. Uno de los motivos fue por falta de tiempo y otro

por que a los usuarios del segundo grupo no conseguimos captar su completa atención. A continuación, se explicará detalladamente el resultado obtenido en cada parte del formulario.

- Búsqueda con nivel fácil: La primera palabra introducida fue "puente" con un nivel de búsqueda fácil y con las opciones de pictograma y definición y ejemplo predeterminadas. En la Figura 4.35 se pueden ver los resultados obtenidos, donde los alumnos comprendían tanto el resultado "construcción" como "estructura" . Y nos comentaron que les pareció bastante útil puesto que les ayudaba aún más a comprender el significado del concepto buscado.

Figura 4.35: Resultado obtenido para la palabra "Puente" con un nivel de búsqueda fácil

- Búsqueda con nivel medio: Se introdujo la misma palabra "puente" pero aumentando el nivel de búsqueda a medio. En la Figura 4.36 se pueden ver los resultados obtenidos. En este caso hubo que explicar a los alumnos los dos nuevos resultados. Por un lado, para la palabra "arco" no entendían la relación existente de este con la palabra buscada y por otro lado, para la palabra "circuito" relacionaban dicho concepto con un circuito de carreras en vez de un circuito electrónico, siendo este último el significado obtenido por la aplicación.

- Búsqueda con nivel avanzado: Esta vez, se volvió a introducir la palabra "puente" con un nivel de búsqueda avanzado. En la Figura 4.37 se pueden ver los resultados obtenidos donde se puede ver que con este nivel se muestra un resultado más que en el anterior nivel. La palabra "placa" puede tener diversos significados según el contexto en el que se utilice, por lo que para los alumnos era complicado entender la relación de "placa" con "puente". Una vez explicado, los alumnos entendieron perfectamente su significado.

- Obteniendo un único resultado: La palabra introducida fue "dictador" con un nivel de búsqueda avanzado y sin mostrar ni pictogramas ni definición y ejemplo. En la Figura 4.38 se pueden ver los resultados

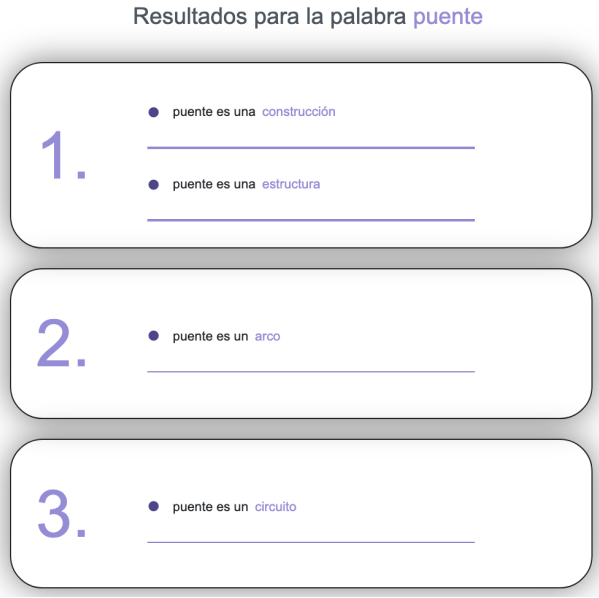

Figura 4.36: Resultado obtenido para la palabra "Puente" con un nivel de búsqueda medio

obtenidos, de estos comprendieron el resultado "lider" pero no sabían que era "soberano". De los demás resultados obtenidos, entendían "gobernador" pero no sabían la diferencia que existía entre este y "gobernante", lo que les llevo a confusión no dejando claro el significado del concepto.

- Obteniendo varios resultados (I): Se introdujo la palabra "defensa" con un nivel de búsqueda fácil. En la Figura 4.39 se pueden ver los resultados obtenidos donde los alumnos entendían 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, ya que les ayuda a entender el significado según el contexto en el que se está utilizando.
- Obteniendo varios resultados (II): Se realizó otra prueba para obtener varios resultados. En este caso de introdujo la palabra "danza" sin pictogramas y con un nivel de búsqueda avanzado. En la Figura 4.40 se pueden ver los resultados obtenidos, de los cuales los alumnos entendieron todos sin ningún problema.
- Obtención de metáforas: Se introdujo la palabra "portero" con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas. Se puede ver en la Figura 4.41 los resultados obtenidos. Todos fueron correctos excepto para el resultado "funcionario" y "administrador", que no entendían su significado. Después, se cambio el nivel de búsqueda a

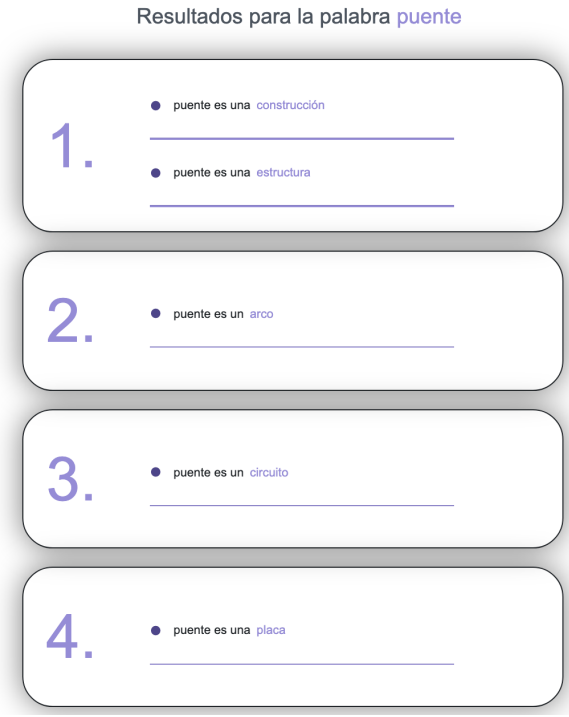

Figura 4.37: Resultado obtenido para la palabra "Puente" con un nivel de búsqueda avanzado

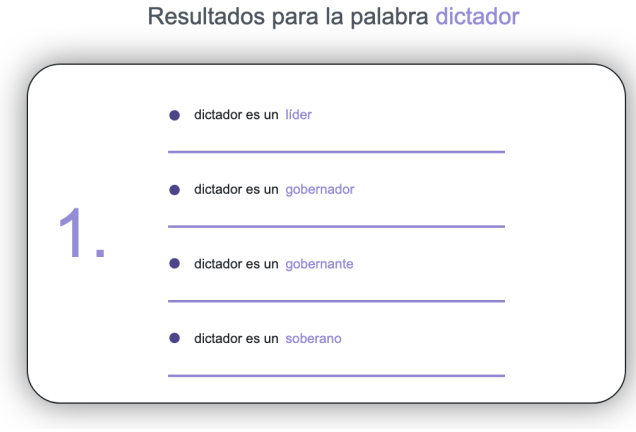

Figura 4.38: Resultado obtenido para la palabra "Dictador" con un nivel de búsqueda avanzado

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: "Portero es un guardia" y "Portero es un funcionario"

Figura 4.39: Resultado obtenido para la palabra "Defensa" con un nivel de búsqueda fácil

de los cuales el primero lo entendieron y el segundo solamente si se añade la opción de mostrar pictograma.

- Obtención de símiles: Se introdujo la palabra "funcionario" con un nivel de búsqueda fácil. Se puede ver en la Figura 4.42 los resultados obtenidos, donde todos fueron comprendidos perfectamente por los alumnos.

- Definición y ejemplo: Se buscó la palabra "familia" con un nivel de

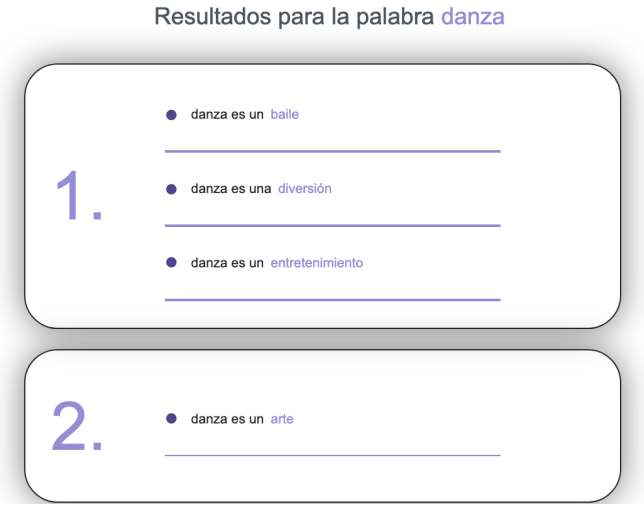

Figura 4.40: Resultado obtenido para la palabra "Danza" con un nivel de búsqueda avanzado

búsqueda fácil y añadiendo la opción de mostrar definición y ejemplo. Se puede ver en la Figura 4.43 los resultados obtenidos, De estos, en la primera ficha no entienden el resultado "clase" pero si les ayuda la definición *("Grupo de cosas que están juntas o que se puede considerar como un todo")*. Igual ocurre en la ficha tres pero con el resultado "línea" *("Descendientes de un individuo")*, donde ellos entienden línea como una recta, pero leyendo la definición comprenden que se refiere a línea sucesoria.

- Uso de mayúsculas: Con la misma palabra "familia", se realizó el cambio de minúsculas a mayúsculas como se puede ver en la Figura 4.44, los alumnos nos comentaron que prefieren esta opción puesto que están más familiarizados y es como aprenden a escribir.

Por último, se llevo a cabo la sección de uso libre, donde los alumnos nos decían palabras para buscar y posteriormente nos daban su opinión sobre los resultados. Las palabras introducidas fueron las siguientes:

- La primera palabra que dijeron fue "pasado" con un nivel de búsqueda fácil. Los resultados obtenidos se pueden ver en la Figura 4.45. Los alumnos nos explicaron que los resultados obtenidos los entendían perfectamente.

- La siguiente palabra que dijeron fue "enamorarse" con un nivel de búsqueda fácil. 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" con un nivel de búsqueda avanzado y

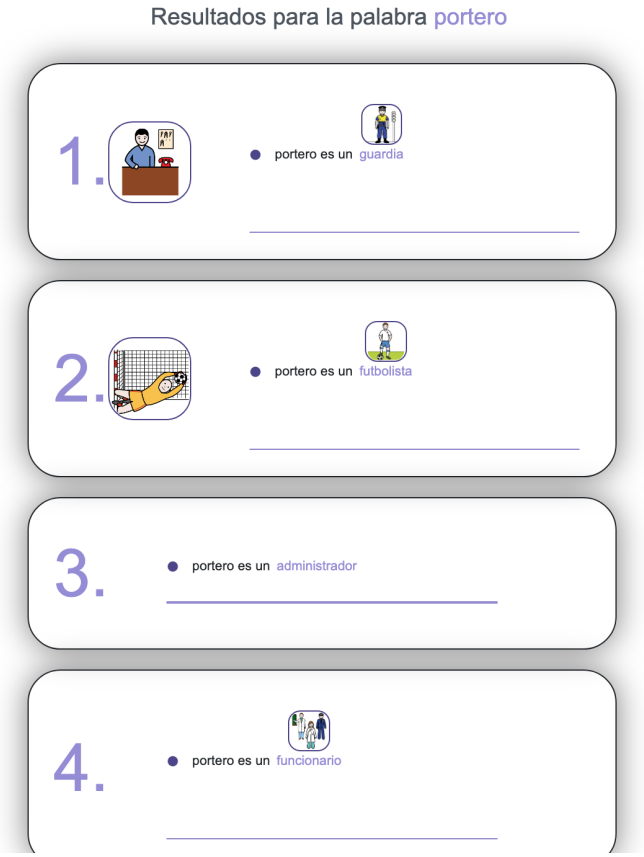

Figura 4.41: Resultado obtenido para la palabra "Portero" con un nivel de búsqueda avanzado y mostrando pictogramas

añadiendo la opción de mostrar pictogramas. El resultado obtenido se puede ver en la Figura 4.46 y gracias al pictograma en el resultado "atraer" pudieron comprender su significado sin necesidad de explicárselo.

- Otra palabra que se buscó fue "fiesta" con un nivel de búsqueda avanzado añadiendo la opción de mostrar definición y ejemplo. Los resultados obtenidos se pueden ver en la Figura 4.47. En la ficha número 4 aparece la palabra "suceso", la cual a los alumnos les confunde un poco. Por ello, decidimos comprobar si la definición y el ejemplo que aparecía les podría ayudar (*"Una ocasión en la que la gente puede reunirse para la interacción y el entretenimiento social", "Un evento social especificado vagamente", "Él planeo una fiesta para celebrar el Día de la Bastilla"*), pero esta definiciones y ejemplo llevan a un mayor grado de confusión al utilizar palabras como "interacción", "vagamente" o "Día de la Bastilla", términos con los cuales no están familiarizados y no conocen sus significados.

Figura 4.42: Resultado obtenido para la palabra "Funcionario" con un nivel de búsqueda fácil

En cambio, la definición que se muestra en la ficha número 2 (*"Cualquier tipo de diversión que produce alegría"*), la entendían perfectamente al utilizar palabras que conocen su significado.

- A continuación, se introdujo la palabra "detective" con un nivel de búsqueda avanzado sin mostrar pictogramas ni definición y ejemplo. Los resultados obtenidos se pueden ver en la Figura 4.48. El resultado de la primera ficha ("Detective es como un agente") la entendían sin ningún problema, pero en la segunda ficha aparece el resultado "Detective es un paco", el cual hace referencia a una obra de teatro y los alumnos no entienden la relación existente entre ese término y el buscado. De los resultados obtenidos, se pinchó en uno de ellos para realizar la búsqueda de ese término en concreto en vez de tener que introducirlo en el buscador. La palabra pinchada fue "investigador" y de los resultados obtenidos, los alumnos entendieron todos los significados.

- La siguiente palabra que se busco fue "policía" con un nivel de búsqueda avanzado sin mostrar pictogramas ni definición y ejemplo. Los resultados obtenidos se pueden ver en la Figura 4.49 y de la primera ficha sabían el significado de todos los conceptos excepto de "dotación" y de la segunda ficha no entendían la relación existente entre el concepto buscado y el resultado "tira". Posteriormente se les explicó que "tira" se refería al verbo tirar.

- Después se introdujo la palabra "sirena" con un nivel de búsqueda avanzado y mostrando pictogramas. Los resultados obtenidos se pueden ver en la Figura 4.50, de los cuales entendían todos pero echaban en falta el resultado de sirena relacionado con el ser mitológico.

Figura 4.43: Resultado obtenido para la palabra "Familia" con un nivel de búsqueda fácil y añadiendo la opción "Mostrar definición y ejemplo"

- Por último, se introdujo la palabra "soso" con un nivel de búsqueda avanzado y mostrando pictogramas. Los resultados obtenidos se pueden ver en la Figura 4.51, y fue correcto excepto que el pictograma de "pesado" en el cual aparece comida, no tiene relación directa con el propio significado de la palabra buscada.

Después de realizar la evaluación guiada, les hicimos unas preguntas relacionadas con la usabilidad de la aplicación. Les preguntamos si la utilizarían habitualmente en su día a día y todos nos contestaron que sí, argumentando que les sirve para poder explicar y entender conceptos más complejos. Otra pregunta que se les realizó fue que si las opciones de mostrar o no tanto los pictogramas, como la definición y el ejemplo y convertir a mayúsculas o minúsculas preferían que fuese configurable o no. La respuesta de todos ellos es que es mucho más cómodo que sea configurable para que de esta forma pueden elegir como mostrar los resultados del concepto buscado. También se les preguntó si les gustaba el color de la aplicación y si les había resultado

Figura 4.44: Resultado obtenido para la palabra "Familia" convirtiendo el texto a mayúsculas

fácil utilizarla, a lo que ellos nos contestaron que si a ambas preguntas. Por último, se les preguntó si preferían los pictogramas a color o en blanco y negro y nos comentaron que a color por que son los que más familiarizados están y se comprenden mejor. En definitiva, a todos los allí presentes les encantó la aplicación así como su funcionalidad.

Tras terminar la evaluación con el primer grupo, se realizó con el segundo grupo. En esta evaluación, 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 analizó con el profesor en ese momento presente. Los resultados obtenidos se explican a continuación:

- Búsqueda con nivel fácil: La primera palabra introducida fue "puente" con un nivel de búsqueda fácil y con las opciones de pictograma y definición y ejemplo predeterminadas. En la Figura 4.35 se pueden ver

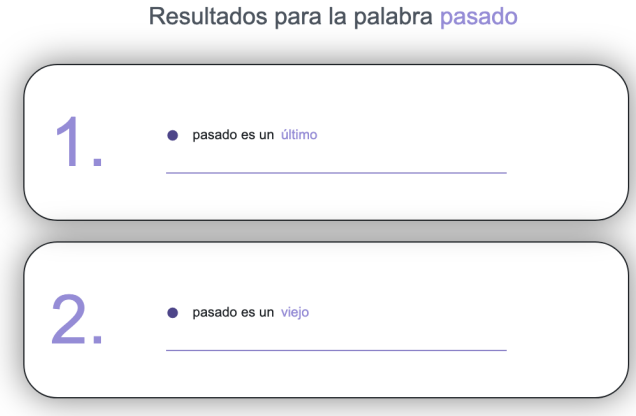

Figura 4.45: Resultado obtenido para la palabra "Pasado" con un nivel de búsqueda fácil

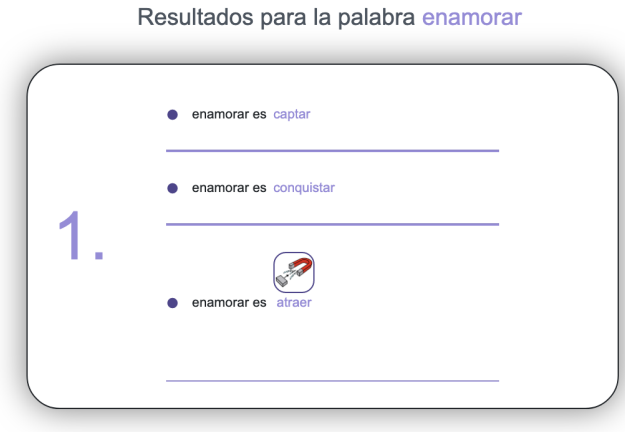

Figura 4.46: Resultado obtenido para la palabra "Enamorar" con un nivel de búsqueda avanzado y añadiendo pictogramas

los resultados obtenidos, donde a los alumnos no les quedaba claro los distintos significados, asi que probamos a añadir la opción de mostrar pictogramas y de esta forma si los comprendían.

- Búsqueda con nivel medio: Se introdujo la misma palabra "puente" pero aumentando el nivel de búsqueda a medio. En la Figura 4.36 se pueden ver los resultados obtenidos. En este caso hubo ocurrió lo mismo que en la prueba 1, y es que si se añaden los pictogramas entienden mucho mejor los resultados que sin ellos, por ejemplo para la palabra "arco".

- Búsqueda con nivel avanzado: Esta vez, se volvió a introducir la palabra "puente" con un nivel de búsqueda avanzado. En la Figura 4.37 se

pueden ver los resultados obtenidos donde se puede ver que con este nivel se muestra un resultado más que en el anterior nivel. A los alumnos les ocurrió lo mismo que al grupo 1 con la palabra "placa", ya que no entendían su significado en dicho contexto.

- Obteniendo un único resultado: La palabra introducida fue "dictador" con un nivel de búsqueda avanzado y sin mostrar ni pictogramas ni definición y ejemplo. En la Figura 4.38 se pueden ver los resultados obtenidos, de estos no entendían el significado de la palabra "soberano", así que se modificó el nivel a medio donde comprendían mejor los resultados. En general, ellos entendían la palabra "dictador" como persona que dicta un texto a alguien, por lo que a través de la aplicación supieron nuevas acepciones para dicho concepto.

- Obteniendo varios resultados (I): Se introdujo la palabra "defensa" con un nivel de búsqueda fácil. En la Figura 4.39 se pueden ver los resultados obtenidos. Estos resultados eran bastante complicados de entender para los alumnos. En la dicha número 1, les confundía la palabra "construcción", aunque explicándoselo si sabían lo que era.

- Obteniendo varios resultados (II): Se realizó otra prueba para obtener varios resultados. En este caso de introdujo la palabra "danza" sin pictogramas y con un nivel de búsqueda avanzado. En la Figura 4.40 se pueden ver los resultados obtenidos, de los cuales los alumnos entendieron todos sin ningún problema.

- Obtención de metáforas: Se introdujo la palabra "portero" con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas. Se puede ver en la Figura 4.41 los resultados obtenidos. Los alumnos conocían los significados que están relacionados con portero de deporte y portero de profesión, pero la palabra "funcionario" y "administrador" no la entendían. A parte, echaban en falta que apareciese el resultado de portero automático.

A partir de este momento, la prueba se realizó con el profesor. Nos comentó que en su día a día al impartir clase existían ciertas palabras que no sabe como explicar su significado a los alumnos. Una de ellas fue "talar", la cuál la buscamos en la aplicación con nivel de búsqueda avanzado y añadiendo pictogramas. El resultado obtenido se puede ver en la Figura 4.52 y este era correcto, pero nos comentó que faltaba el pictograma que representa el concepto en sí. Del resultado que se obtuvo ("talar es cortar"), el pictograma de "cortar" no era el adecuado según su acepción, ya que debería aparecer uno de "talar" en vez de "cortar".

Otra palabra que se buscó fue "sacudir" con nivel de búsqueda avanzado y mostrando tanto los pictogramas como la definición y el ejemplo. Los

resultados obtenidos se pueden ver en la Figura 4.53 . Algunos de estos fueron correctos y otros no. De los correctos, el profesor indicó que debería aparecer primero el resultado más común o el que 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. Por otro lado, la definición y ejemplo que aparecen son correctos (*Mover o causar movimiento de un lado a otro*).

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

4.3.3. Análisis de los resultados

Finalmente, tras realizar la evaluación con ambos grupos y teniendo ya resultados firmes nos dimos cuenta de la gran diferencia que existía entre la utilización de la aplicación por parte del primer grupo y del segundo.

La mayoría de los resultados obtenidos en el primer grupo fueron entendidos sin ningún problema. En el caso de la búsqueda de palabras polisémicas, conocían el significado de gran parte de los resultados obtenidos e incluso echaban en falta que apareciesen más conceptos relacionados con la palabra buscada. En cambio, en el segundo grupo eran muy pocos los resultados que comprendían sin necesidad de explicárselo, estos son conceptos bastante complicados para ellos, haciendo así que les lleve a más confusión en vez de aclararles y facilitarles la comprensión del significado.

Por otro lado, el orden en que se muestran los resultados para el primer grupo es correcto pero en cambio para el segundo grupo no. Para ellos debería aparecer primero el resultado más común y directo con la palabra búsqueda.

En cuanto a los pictogramas, a ambos grupos les resultaba necesario su aparición. Haciendo así que un resultado que no comprendían, gracias a estos podían hacerse una idea cercana de su significado. Hay que destacar que para el segundo grupo, los pictogramas mostrados deberían ser aún mas aclaratorios, es decir, debería aparecer el pictograma resultante de la relación existente entre el resultado y el contexto en el que se utiliza y no el pictograma de la palabra en sí.

Sobre la definición y el ejemplo, la gran mayoría de resultados no sirve para ayudar a entender el significado, puesto que son frases complejas, muy largas y algunas no tienen una relación directa con el resultado en sí. Sobre todo para el segundo grupo, las definiciones y ejemplos mostrados les confundía más ya que no se llegaba a entender ni la definición ni el ejemplo de la mayoría de las palabras buscadas.

Por último, la diferencia entre leer el texto en mayúsculas y minúsculas era bastante considerable. Un texto en mayúsculas les ayudaba a leer con mayor facilidad.

Tras estos resultados, obtuvimos distintas conclusiones. La principal es

que la aplicación solo es útil para usuarios cuyo grado de discapacidad cognitivo no fuera muy elevado. Puesto que si la aplicación fuera destinada para personas como las del grupo 2 no les ayudaría, si no todo lo contrario, les complicaría aún más el poder entender el significado de cierto concepto. Sin embargo, para personas con un grado de discapacidad menor les sería de gran utilidad. Ellos necesitan resultados mucho más fáciles de los que la aplicación devuelve y organizados en una estructura totalmente distinta a la que nosotros implementamos. En cambio, para personas con un mayor grado de independencia les resultaría bastante útil y les simplificaría ciertas situaciones que puedan vivir.

Por otro lado, la aplicación debería mostrar los pictogramas según la relación existente entre el resultado obtenido y la palabra buscada y no de la palabra en sí. De esta forma, a los usuarios les sería de gran utilidad. Igualmente deberían aparecer pictogramas en todos los resultados y no solo en algunos.

Las definiciones y ejemplos que se muestran no son de gran utilidad, deberían aparecer resultados mucho más sencillos, con frases mucho más cortas y sin utilizar palabras tan complejas.

La aplicación en sí no es precisa en cuanto a los resultados obtenidos, muchos de ellos no tienen relación directa con el concepto buscado por lo que devuelve datos erróneos. Pero en cambio, para otras palabras devuelve resultados claros y precisos haciendo que su entendimiento sea mucho más fácil y rápido.

Figura 4.47: Resultado obtenido para la palabra “Fiesta” con un nivel de búsqueda avanzado y añadiendo tanto pictogramas como definición y ejemplo

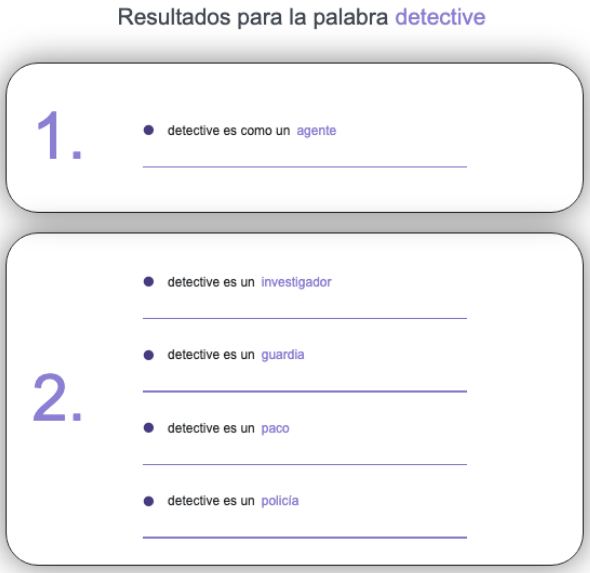

Figura 4.48: Resultado obtenido para la palabra “Detective” con un nivel de búsqueda avanzado

Figura 4.49: Resultado obtenido para la palabra "Policía" con un nivel de búsqueda avanzado

Figura 4.50: Resultado obtenido para la palabra “Sirena” con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas

Figura 4.51: Resultado obtenido para la palabra “Soso” con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas

Figura 4.52: Resultado obtenido para la palabra “Talar” con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas

Figura 4.53: Resultado obtenido para la palabra "Sacudir" con un nivel de búsqueda avanzado y añadiendo la opción de mostrar pictogramas y de definición y ejemplo

Capítulo 5

Trabajo Realizado

En este capítulo se describe el trabajo realizado por cada uno de los autores de este proyecto.

5.1. Trabajo realizado por Irene

Al comenzar el proyecto, la primera tarea a realizar consistió en investigar las bibliotecas que se podrían utilizar para el procesado de las palabras. Al principio se encontró una biblioteca para el procesado de texto en Python llamada *NLTK*, pero se pudo comprobar que las etiquetas que asignaba a las palabras no eran del todo correctas. Se decidió buscar otra biblioteca distinta y se recurrió a Spacy. Con esta librería ya se pudo etiquetar correctamente todas las palabras, lo que dio lugar a un primer diseño de un programa utilizando Jupyter. A continuación, investigué qué tecnologías utilizar para la realización del prototipo tecnológico. Encontramos como entorno de desarrollo *Pycharm* y como framework de desarrollo web *Django*.

Una vez seleccionadas estas tecnologías, se investigó cómo poder sacarles el mejor partido y comenzamos a desarrollar el primer el prototipo tecnológico. En esta tarea, mi cometido principal fue conectar las vistas html con la lógica en Python. A continuación tuvimos que aprender cómo se implementa un formulario web y como se hacía una redirección a la vista. Una vez tuvimos claro cómo realizar esta tarea, integramos el código desarrollado inicialmente en Jupyter en nuestro servicio web, finalizando el prototipo tecnológico.

En cuanto a la memoria, dividimos su elaboración en dos partes iguales entre los dos miembros del grupo, intentando que ambos tuviésemos asignada una parte en todos los capítulos de la misma, por lo que ambos redactamos tanto una parte de la introducción (en la que me correspondió redactar la motivación) como el estado de la cuestión (en este caso me correspondió el apartado de Lectura Fácil y Procesamiento del Lenguaje Natural).

La investigación de como funcionaba ConceptNet y su API la hicimos conjuntamente. A partir de aquí se realizó un prototipo con esta red semántica que contenía dos servicios web uno para los sinónimos y otro para los términos relacionados.

Por otro lado se obtuvieron ficheros con las 1.000, 5.000 y 10.000 palabras más usadas según la RAE, se hizo un filtrado de estas palabras para quedarnos solo con adjetivos, verbos y nombres utilizando SpaCy.

Tras la realización del prototipo y hacer las pruebas correspondientes nos dimos cuenta que los resultados obtenidos no eran válidos por lo que se buscó otra red semántica.

Se decidió probar con WordNet, inicialmente tuvimos muchos problemas para poder obtener los recursos, ya que no tienen una API, sino que es una base de datos que te debes descargar.

Con los recursos descargados, estudiamos como estaban estructurados para poder utilizarlos según nuestros intereses y nos pusimos a programar el prototipo.

Nos descargamos los ficheros de la base de datos e investigamos como funcionaban para poder utilizarlos. Yo empecé a diseñar la vista del prototipo. Y después, nos pusimos los dos con la parte del backend. La implementación de todos los servicios web se realizó conjuntamente entre mi compañero y yo.

Después, ya empezamos a trabajar en lo que sería el proyecto propiamente dicho, empezando por diseñar la vista. Para ello realizamos una iteración competitiva tanto Pablo como yo, para ver con que diseño nos quedábamos finalmente.

Se decidió que mi diseño de la vista era el que se va a implementar, añadiendo algunos aspectos que había añadido Pablo en su diseño. Por lo que adapté mi vista añadiendo dichos cambios para utilizarla en el proyecto que se desarrollaría finalmente.

A continuación, desarrollé la parte de la lógica de la vista de la aplicación, ya que aparte de la lógica de negocio desarrollada en Python, para mostrar los datos correctamente utilizamos jQuery con peticiones ajax para tratar los datos. Esta parte me llevó varios días ya que se presentaron varias dificultades.

Al terminar esta tarea, nos reunimos Pablo y yo para desplegar el proyecto en el contenedor que nos proporcionó el grupo NIL de la facultad de informática.

Cuando subimos el proyecto al contenedor, nos dimos cuenta de que había varios problemas, por lo que estuvimos varios días solucionándolos para que funcionase correctamente de cara a la evaluación que teníamos por delante en pocos días.

Para la evaluación de la aplicación, creamos entre Pablo y yo un formulario de Google Forms. Y una vez realizada, se transcribió a la memoria también entre los dos.

En todo momento, a la vez que se realizaba la implementación de código (ya sea backend o fronted) se iba realizando la memoria. Siempre dividida a partes iguales entre mi compañero y yo.

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.

Tras investigar el funcionamiento de esta red semántica, empezamos a implementar un prototipo para probar si es funcional.

Mientras estábamos trabajando en esto, avanzamos en la redacción de la memoria, dividiéndonos el trabajo equitativamente.

Cuando terminamos la implementación del prototipo, se lo presentamos a los directores del TFG y nos dimos cuenta de que no iba a ser útil por lo que empezamos a trabajar en otra red semántica: WordNet.

Investigamos como funcionaba esta red semántica, estuvimos varios días con esto, hasta que descubrimos para acceder a los datos no hay una API pública sino que es una base de datos que hay que descargarse.

Encontramos varios sitios que hacían uso de la misma y para poder descargarse los recursos había que registrarse. Me registré en varios de ellos pero no nos servían ya que no estaban en castellano. Finalmente tras varios días encontramos los recursos necesarios en la página web de la universidad de Princeton.

Con los recursos descargados, estudiamos como estaban estructurados para poder utilizarlos según nuestros intereses y nos pusimos a programar el prototipo.

Mientras Irene empezaba a diseñar la vista, yo empecé a desarrollar la lógica interna para integrarla con los recursos descargados.

Después nos pusimos los dos con la parte de la lógica del prototipo para terminarlo cuanto antes.

A continuación, empecé a diseñar una prueba estadística para comprobar de manera empírica que red semántica ofrece mejores resultados. Realicé la prueba y la puse en común con Irene para analizar los resultados obtenidos.

Con los resultados de las pruebas y tras enseñar a los directores del TFG los resultados que ofrecía, decidimos que con estos recursos se desarrollaría la aplicación.

Empezamos a realizar la aplicación final. Me puse a trabajar en mi prototipo visual para la iteración competitiva con Irene para ver como sería el diseño visual.

A continuación, empecé a desarrollar la base de datos de pictogramas para integrarlos en nuestra aplicación. Me descargué los pictogramas e implementé un script que relacionaba los pictogramas descargados con los recursos de WordNet, utilizando la API pública de ARASAAC, que relacionaba sus pictogramas con los recursos de WordNet mediante el offset. Sin embargo, estos offset correspondían a una versión de WordNet que no era la que estábamos utilizando, por lo que tuve que utilizar una API de la universidad de Princeton para traducir dichos offset a los que utiliza la versión que trabajamos nosotros, que es la 3.0.

Cuando acabé la base de datos de pictogramas, Irene y yo subimos la aplicación que teníamos implementada en local al servidor ofrecido por el grupo NIL de la facultad de informática para que pudiese estar disponible para todo el mundo.

Cuando subimos el proyecto, nos dimos cuenta de que en el servidor había varios problemas, por lo que estuvimos varios días solucionándolos para que funcionase correctamente de cara a la evaluación que teníamos por delante en pocos días.

Una vez subido y con los errores corregidos, seguimos avanzando en el desarrollo de la memoria y comenzamos a diseñar un formulario para realizar la evaluación de la aplicación con usuarios finales terminando el desarrollo del proyecto.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

Tras finalizar el proyecto y explicar todo lo recogido en esta memoria, se han podido recabar una serie de conclusiones atendiendo a los objetivos planteados y como se han conseguido.

Como se ha explicado desde el inicio de este trabajo, existen ciertos colectivos en nuestra sociedad que sufren de discapacidad cognitiva, haciendo que el significado de ciertas palabras no puedan ser entendidos. Esto les afecta directamente en su vida personal y profesional, ya que supone una gran limitación para ellos. El objetivo principal del presente trabajo, era crear una aplicación web basada en servicios web que dada una palabra compleja para el usuario, devuelva mediante el uso de figuras retóricas (metáforas, símiles y analogías) un concepto mucho más sencillo.

Definir el concepto de sencillo es bastante relativo según para que persona, ya que no existe un grado definido de que es una palabra fácil. Por lo que, para la realización de este trabajo se tomó como referencia los listados facilitados por la RAE de las 1.000, 5.000 y 10.000 palabras más usadas del castellano.

La implementación de cada servicio web se ha hecho de manera modular y están accesibles en una API pública, para que de esta forma puedan ser reutilizados en futuras aplicaciones por distintos desarrolladores.

Respecto a la creación de la interfaz, se han utilizado colores que resaltan sobre fondos claros, así como intentando que esta fuese lo más juvenil posible. En especial, hay que recalcar que gracias a la opinión de expertos que trabajan en centros dedicados a la educación de este colectivo, nos han ayudado a que elementos deberían aparecer y de que manera, así como se deberían mostrar los resultados, entre otros aspectos.

Para crear una aplicación que fuese aún más útil, se añadieron una serie de opciones configurables, consiguiendo de esta forma que los usuarios

pudieran decidir como quieren obtener los resultados. Estas opciones configurables son:

- Poder convertir todo el texto de la página en mayúsculas: Esto facilita la lectura puesto que están familiarizados a leer en mayúscula y es como aprenden a escribir.
- Poder mostrar pictogramas o no: Gracias a la utilización de pictogramas, los conceptos pueden ser entendidos perfectamente.
- Poder mostrar una definición y ejemplo del resultado: Por si el pictograma no terminara de ayudar al usuario a comprender el concepto, decidimos añadir una definición y un ejemplo para ayudarles aún más en su comprensión.

Cabe destacar que la aplicación se construyó desde cero, sin hacer uso de plantillas externas. La aplicación web ha sido sometida a una evaluación con usuarios finales, donde pudimos probar tanto su funcionalidad como su interfaz. Tras esta evaluación pudimos comprobar que la aplicación solamente es útil para personas con problemas semánticos no muy altos y surgieron bastantes aspectos que mejorar.

Este trabajo ha sido una oportunidad para aplicar todos los conocimientos adquiridos en distintas asignaturas cursadas durante la carrera a un proyecto grande y de impacto real. Entre estas asignaturas cabe destacar:

- **Estructura de Datos y Algoritmos** para ayudarnos a pensar en el código de una manera eficiente y estructurada, así como a elegir las estructuras de datos más adecuadas.
- **Fundamentos de la Programación** y **Tecnología de la Programación** para las base de la lógica interna de la aplicación.
- **Aplicaciones Web** e **Interfaces de Usuario** para obtener los conocimientos suficientes de HTML, CSS, Bootstrap, jQuery y JavaScript para desarrollar la aplicación así como el funcionamiento básico de una aplicación web y las reglas básicas que debe contener una interfaz.
- **Base de Datos** y **Ampliación de Base de Datos** para el tratamiento y gestión de múltiples datos.
- **Gestión de Proyectos Software** para la gestión adecuada de la aplicación.
- **Administración de Sistemas y Redes** que nos ayudo a la configuración del servidor así como al despliegue de la aplicación en el mismo.
- **Ingeniería del Software** y **Modelado del Software** para la realización de la parte de modelado y los respectivos diagramas.

- **Ética, Legislación y Profesión** para obtener los conocimientos sobre licencias, tanto para nuestro propio código como para tratar y utilizar el código de terceros.

Por otro lado, también hemos aprendido bastantes cosas nuevas, como puede ser Django, Python, Latex, configurar un servidor y Procesamiento de Lenguaje Natural. Ambos integrantes del equipo hemos aprendido sobre las discapacidades cognitivas existentes, y gracias a la evaluación de la aplicación pudimos convivir unas horas con ellos, haciendo que fuese una experiencia muy enriquecedora en lo personal ya que pudimos comprobar las dificultades que afrontan en su día a día y como se enfrentan a ellas con positivismo y naturalidad.

En definitiva, de los objetivos que se plantearon en la sección 1.2 se han cumplido la gran mayoría, y los que no se pudieron cumplir se explicarán en la siguiente apartado.

6.2. Trabajo Futuro

Como se ha comentado en el apartado anterior, no se han podido cumplir todos los objetivos marcados en un primer momento. El motivo principal de esto es por falta de tiempo, ya que al ser un proyecto tan grande y al tener que cubrir el mayor número de necesidades posible, se convierte en un trabajo aún mayor. Por ello, todos los aspectos que no se han llegado a suplir, tanto los fijados en un primer momento como los sugeridos en la evaluación final se dejarán aquí reflejados como trabajo futuro.

- Mostrar resultado mediante analogías: Al igual que nuestra aplicación muestra los resultados mediante metáforas y símiles, también deberían mostrarse mediante analogías. Por ello, habría que buscar la información necesaria para saber como enlazar el concepto buscado y el resultado mostrado y ver que características comparten ambos.

- Añadir reconocimiento por voz: En cada elemento de la interfaz debería haber una descripción auditiva del elemento seleccionado para facilitar aún más su uso.

- Video explicativo del pictograma: Al lado de cada pictograma debería aparecer un botón que al ser pulsado mostrara un video explicativo del concepto.

- Mostrar primero significado más utilizado: Una vez que el usuario haya buscado una palabra y se muestren los resultados en fichas, dentro de cada una debería aparecer primero el concepto más relacionado con el término buscado.

- Mostrar pictograma con la acepción correcta: En vez de realizar la búsqueda del pictograma de una metáfora o un símil, debería buscarse el pictograma de la relación existente entre el resultado y el concepto buscado.
- Poder buscar palabras más sencillas y obtención de resultados más sencillos: En vez de comparar los resultados obtenidos con las 1.000, 5.000 y 10.000 palabras de la RAE, se debería encontrar la manera de poder obtener resultados aún más sencillos para conceptos no tan complejos.
- Implementación de la aplicación móvil: Sería bastante útil que los usuarios pudieran tener en su teléfono la aplicación para poder acceder a ella siempre que quisieran.

En un futuro próximo nos gustaría poder seguir con este trabajo, ya que nos hemos involucrado a nivel personal con el y nos gustaría que en un futuro pueda ser una aplicación usable por cualquier persona.

Chapter 6

Conclusions and Future Work

6.1. Conclusions

After completing the project, some conclusions have been obtained in relation to the objectives initially proposed.

The main goal of this work was to create a web application, based on web services, in which difficult words could be converted into easier words by using rhetorical figures (metaphors, similes and analogies).

It is difficult to define what an easy concept means because is not the same for everyone. To do so, we have taken the RAE's 1.000, 5.000 and 10.000 most used words in the Spanish language.

The implementation of each web service has been made in a modular way and they are available in a public API, so they can be used by other developers at any point.

When we created our application interface, we used dark colours that stand out on plain backgrounds in order to make reading easy for the users. We consulted with professionals with vast experience in this field so they could give us advice on how to display the results for the words searched.

To make our application more useful we added some customizable settings for the user, such as:

- To be able to convert the whole text into capital letter: this makes reading easier because that is how they learn to read and write.
- To be able to show pictograms if needed: using pictograms makes it easy to understand certain concepts.
- To be able to show definitons and examples for the results: if showing pictograms is not enough to understand a concept, the application can show them an example and definition.

We built this application from scratch, not using any external templates. This web application was tested by target users so we could prove its functionality and its interface. As we concluded these tests, we realized the application is useful for people with mild disabilities but it needs to be improved in order to make it useful for people with severe disabilities.

This project has been an opportunity to apply all the knowledge acquired in different subjects taken during the degree to a large project with real impact. Among all of these subjects, we can highlight:

- Data Structures and Algorithms, that helped us to acquire a structured and efficient way of thinking when programming.
- Foundations of the Programming and Technology of the Programming, which helped us understand the internal logic of the applications.
- Web Applications and User Interfaces, that helped us to acquire enough knowledge about html, css, Bootstrap, jQuery and javascript to develop the application and the basic rules about interface design.
- Databases and Extension of Databases to manage big volumes of data.
- Software Project Management, for the correct management of our application.
- System and Networks Management, that helped us to the configuration of our server.
- Software Engineering and Software Modeling, that helped us to design the application.
- Ethics, Legislation and Profession, where we learned about software licenses, both to protect our code and to know how to correctly use free software developed by third parties.

We have also acquired knowledge about some new things such us Django, Python, Latex and Natural Laguage Processing. We have learned about cognitive disabilities and we have got to spend some time with target users, which made this experience a really valuable one. We have realized the problems they face and the way they surpass them.

We consider we have achieved most of our main goals (section 1.2). The ones which could not be achieve will be explained in the next section.

6.2. Future Work

We couldn't achieve every goal we aimed to at the beginning of the project. The main reason is the lack of time and the fact that we needed to cover so many fields. The items that can be developed as future work are:

- To show results with analogies: Our application shows results with metaphors and similes. We think it would be useful for the user to show them analogies too. For that purpose, we would need to search information about how to link the concept and the result and which properties they share.
- To add voice synthesis: We should add an audio description of the selected item to make the application easier for the user.
- Explanatory video of each pictogram: a button could be added that, when clicked, shows an explanatory video.
- To show the most relevant result first: When the user has searched for a word, the most significant results should be shown first. This has been difficult to achieve, since the external resources used to mine for words do not provide any information that allows to decide what concepts are more relevant.
- The pictograms should be displayed with the correct meaning: in case of polysemic words, the pictogram should be shown with the correct relation between the result and the searched concept.
- We should be able to look for easier words and easier results: we need to find a way to obtain easier results rather than using RAE's 1.000, 5.000 and 10.000 lists of easy words.
- Develop a smartphone application: It would be useful for the user to be able to have this application in their smartphone instead of as a web application.

In the near future, we would like to keep working in this project, as during this year we have been involved in its development at a personal level, so we would like it to be an application that could be used by everyone.

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 disrupciones 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 desentramar 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, dinstinguido 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 hechos 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 Solskjaer. 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 arrogan 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 trata 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.

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 linguisticos 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*. Bllomsbury 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.