Mejora de la Comprensión Lectora mediante Analogías para la Inclusión

Trabajo de Fin de Grado
Curso 2017–2018

Autor
Irene Martín Berlanga
Pablo García Hernández

Director
Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo

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

Mejora de la Comprensión Lectora mediante Analogías para la Inclusión

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

Autor
Irene Martín Berlanga
Pablo García Hernández

Director
Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo

Dirigida por el Doctor
Virginia Francisco Gilmartín
Gonzalo Rubén Mendez Pozo

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

2 de diciembre de 2018

Autorización de difusión

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

Nombre Del Alumno

2 de diciembre de 2018

Dedicatoria

Texto de la dedicatoria...

Agradecimientos

Texto de los agradecimientos

Resumen

Resumen en español del trabajo

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Abstract in English.

Keywords

10 keywords max., separated by commas.

Índice

1. Introduction **1**

1. Introducción **3**
1.1. Motivación 3
1.2. Objetivos 4
1.3. Estructura de la memoria 5

2. Estado de la Cuestión **7**
2.1. Lectura Fácil 7
2.2. Procesamiento del Lenguaje Natural 9
2.2.1. ConceptNet 10
2.2.2. Alternativas a conceptnet 13
2.3. Figuras retóricas 13
2.4. Servicios Web 14
2.4.1. Arquitectura Servicios Web 15
2.4.2. Ventajas de los Servicios Web 16
2.4.3. Desventajas de los Servicios Web 16

3. Trabajo Realizado **21**
3.1. Trabajo realizado por Irene 21
3.2. Trabajo realizado por Pablo 21

4. Conclusiones y Trabajo Futuro **25**

4. Conclusions and Future Work **27**

A. Título **29**

B. Título **31**

Índice de figuras

2.1. Logo Lectura Fácil 8
2.2. Ejemplo Red Semántica 10
2.3. Página principal Concepnet 11
2.4. Resultados de búsqueda de una palabra 12
2.5. Vista de JSON 18
2.6. Vista de JSON 19

Índice de tablas

Chapter 1

Introduction

Introduction to the subject area.

Capítulo 1

Introducción

En el capítulo 1 vamos a realizar una introducción a nuestro TFG. Dicha introducción la dividiremos en varias secciones. En el apartado 1.1 hablaremos de la razón de ser nuestro trabajo, en el 1.2 de los objetivos que queremos alcanzar con el mismo y en la 1.3 de la estructura de esta memoria.

1.1. Motivación

En nuestra sociedad actual, existen ciertos colectivos como pueden ser las personas con algún tipo de trastorno cognitivo, inmigrantes, ancianos, analfabetos funcionales, niños, etc... que tienen dificultad para aprender conceptos complejos o no tan complejos. Esto supone una serie de limitaciones en su vida cotidiana, en la forma de relacionarse con otros individuos, en su vida profesional e incluso su vida personal.
Existen multitud de palabras cuyo significado es bastante complicado de explicar de una manera sencilla, por lo que una solución para que cualquier persona lo pueda comprender es hacer uso de metáforas o analogías. De esta forma, se puede asimilar el concepto de una manera más rápida y sencilla.

Por ejemplo, suponemos que un inmigrante necesita realizar gestiones para legalizar su estancia o firmar documentación. Si no tiene una buena comprensión del lenguaje no sabe las consecuencias de firmar los documentos porque no los entiende. Otro ejemplo que podríamos añadir sería cuando una persona firma un contrato, como por ejemplo, una póliza de seguros o un contrato de trabajo, en el cual añaden ciertas cláusulas que son incomprensibles.
Para ayudar principalmente a estas personas a que puedan entender el significado de cualquier palabra, y de esta forma superar algunas de sus limitaciones, vamos a desarrollar una aplicación que permita definir palabras complejas mediante comparaciones con otras más fáciles ya conocidas por

los usuarios. Por ejemplo, si queremos explicar una palabra compleja como puede ser piraña, podemos describir sus características utilizando conceptos simples para facilitar su entendimiento de la siguiente manera:

Una piraña nada como un pez y es agresiva como un león

Mediante esta comparación, alguien que desconozca completamente el significado de *piraña*, puede hacerse una idea muy aproximada de lo que es.

1.2. Objetivos

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

- Desarrollar una aplicación que esté al alcance de todos los usuarios, es decir, ninguna persona puede quedarse excluida de su uso por ningún motivo, debe ser una aplicación sencilla, fácil de usar y entender.
- La aplicación estará construida con servicios web que la doten de funcionalidad.
- La aplicación funcione con un amplio número de palabras complejas para que tenga la mayor utilidad posible.
- Los servicios web desarrollados estarán disponibles en una API pública para que cualquiera pueda utilizarlos.
- Construir la aplicación de manera incremental, añadiéndole valor al producto poco a poco.
- Construir una aplicación que esté centrada en el usuario, que sea lo más amigable posible y que solucione problemas reales de una forma usable.
- La aplicación tenga una apariencia atractiva y amigable con el usuario.

Alcanzando los objetivos anteriormente descritos, conseguiremos obtener un producto de calidad, con una gran utilidad tanto social como académica, que

puede ayudar a mucha gente a aprender nuestro idioma de una manera más fácil.

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

1.3. Estructura de la memoria

La memoria se dividirá en varios capítulos. El contenido de los mismos es el siguiente:

- El **capítulo uno** (capítulo actual) contiene la introducción.
- El **capítulo dos** es el estado de la cuestión, en el que se explicará que es la lectura fácil y como se aplica y se introducirán los conceptos de Procesamiento del Lenguaje Natural(PLN), figuras retóricas y servicios web.
- El **capítulo tres** describe el trabajo realizado por cada uno de los autores.

Capítulo 2

Estado de la Cuestión

En este capítulo se van a tratar aspectos importantes dentro del ámbito de la retórica así como una explicación detallada de lo que es la lectura fácil, sin olvidar aquellas herramientas y tecnologías que se van a utilizar. En la sección 2.1 se explica el concepto de lectura fácil, su historia, pautas a seguir y ejemplos para que se pueda entender aún mejor. En la sección 2.2 hablaremos del procesamiento del lenguaje natural, y la relación que tiene con nuestro trabajo, en la sección 2.3 figuras retóricas trataremos el tema de las figuras retóricas, que son y que uso le daremos. Por último, en la sección 2.4 se explica el concepto de servicio web, su funcionalidad, ventajas y desventajas y tipos existentes.

2.1. Lectura Fácil

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

La lectura fácil surgió en Suecia en el año 1968, donde se editó el primer libro en la Agencia de Educación en el marco de un proyecto experimental. A continuación, en 1976 se creó en el Ministerio de Justicia un grupo de trabajo para conseguir textos legales más claros. En 1984 nació el primer periódico en lectura fácil, titulado "8 páginas", que tres años más tarde,en 1987, se 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 [1] .

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

Figura 2.1: Logo Lectura Fácil

En la actualidad, podemos distinguir los documentos en lectura fácil gracias al logo de la figura 2.1:

Los documentos escritos en Lectura Fácil son documentos de todo tipo que siguen las directrices internacionales de la IFLA (International Federation of Library Associations and Institutions) y de Inclusion Europe en cuanto al contenido y la forma. Algunas pautas a seguir para escribir correctamente un texto en Lectura Fácil son: (**?**)

- Evitar mayúsculas fuera de la norma, es decir, escribir en mayúsculas sólo cuando toca según las reglas ortográficas, como por ejemplo, después de un punto o la primera letra de los nombres propios.
- Debe evitarse el punto y seguido, el punto y coma y los puntos suspensivos.
- El punto y aparte hará la función del punto y seguido.
- Evitar corchetes y signos ortográficos poco habituales, como por ejemplo: %, & y /.
- Utilizar oraciones simples. Por ejemplo: *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. 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: *perro, casa, gato, coche, comer...*
- Evitar abreviaturas, acrónimos y siglas.
- Alinear el texto.

- 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.
- Evitar frases superiores a 60 caracteres.
- Si el documento está paginado, incluir la paginación claramente y reforzar el mensaje de que la información continúa en la página siguiente.

Estos son solo algunos de los aspectos que se deben tener en cuenta para realizar correctamente un documento en lectura fácil.
Debemos también hacer hincapié en la distinción entre palabras fáciles y complejas, puesto que son de gran importancia para la lectura fácil. Las palabras complejas son aquellas que no se utilizan a menudo en nuestra sociedad, como por ejemplo: *Melifluo* o *inefable*. [2] Es por ello que este tipo de palabras deben estar totalmente descartadas en la lectura fácil, y en su lugar debemos introducir palabras fáciles, que son aquellas que se utilizan asiduamente. La RAE dispone de un documento con las mil palabras más usadas.

2.2. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural(PLN) es una rama de la Inteligencia Artificial que se encarga de investigar la manera de comunicar máquinas con personas mediante el uso de lenguajes naturales (entendiendo como lenguaje natural el idioma usado con fines de comunicación por humanos, ya sea hablado o escrito, como pueden ser el español, el ruso o el inglés). El Procesamiento del Lenguaje Natural se ayuda de las redes semánticas, puesto que estas son una forma de representación del conocimiento lingüístico en la que los conceptos y sus interrelaciones se representan mediante un grafo. Se utilizan para representar mapas conceptuales y mentales (**?**). Los nodos están representados por el elemento lingüístico, y la relación entre los nodos sería la arista, como podemos ver en la figura 2.2 . Existen principalmente tres tipos de redes semánticas [3]:

- 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.

[2] http://historiasmaravillosas122.blogspot.com/2015/07/las-20-palabras-mas-bonitas-del-idioma.html
[3] http://elies.rediris.es/elies9/4-3-2.htm

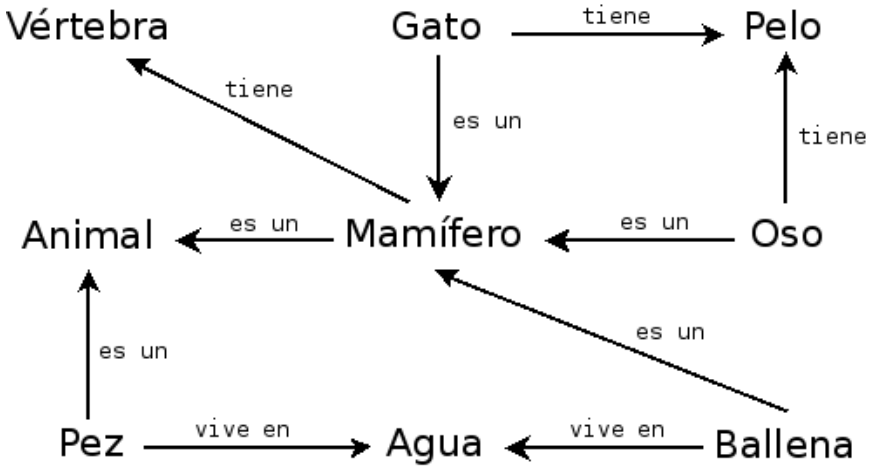

Figura 2.2: Ejemplo Red Semántica

- Grafos Conceptuales: Existen dos tipos de nodos: nodos de conceptos y de relaciones. 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. Es por ello que pueden ser de dos tipos:
  - Nodos de conceptos: Pueden representar una entidad, asi como un estado o proceso.
  - Nodos de relación: Indican como se relacionan los nodos.
- 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.

Para el trabajo que queremos realizar, existen multitud de páginas web que son redes semánticas capaces de procesar el Lenguaje Natural. A continuación, hablaremos de algunas de ellas.

2.2.1. ConceptNet

Es una red semántica creada por el MIT *Massachusetts Institute of Technology* en 1999, diseñada para ayudar a los ordenadores a entender el significado de las palabras. Esta disponible en múltiples idiomas, como el español, el inglés o el chino. ConceptNet dispone de una aplicación web [4] , como se puede ver en la figura 2.3, donde seleccionas el idioma deseado y añades

[4] http://conceptnet.io/

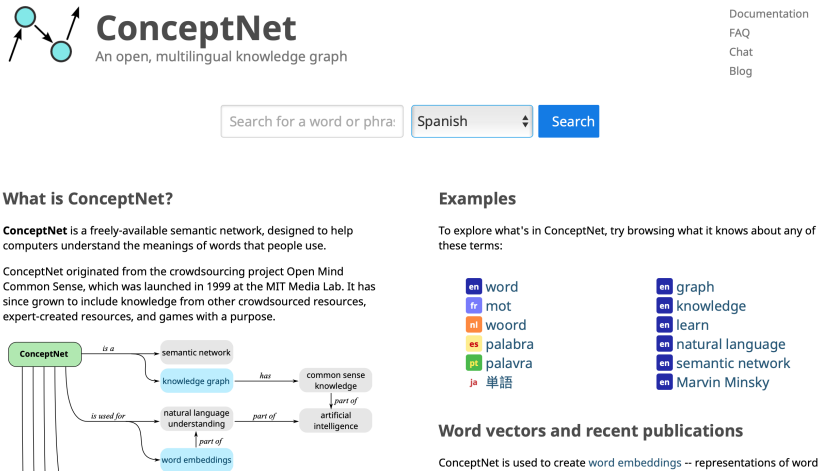

Figura 2.3: Página principal Concepnet

la palabra a buscar. En la figura 2.4, podemos ver que si añadimos la palabra "Pelea"nos devuelve una cantidad de columnas, distinguiéndolas por sinónimos, términos relacionados, símbolos, etc... Principalmente, lo que a nosotros nos interesa son los sinónimos y términos relacionados. Por otro lado, concepnet dispone de un servicio web [5] que devuelve los resultados en formato JSON. Siguiendo con el mismo ejemplo anterior, podemos ver en la figura 2.6 los resultados en dicho formato:

Este consta de cuatro campos principales:

- @context: URL enlazada a un archivo de información del JSON para comprender la API. También puede contener comentarios que pueden ser útiles para el usuario.

- @id: Concepto que se ha buscado y su idioma. En nuestro caso, aparece de la siguiente manera: */c/es/pelea*, donde *c* significa concepto, *es* indica el lenguaje, en este caso, el español y por último *pelea* que es la palabra buscada.

- edges: Que constituyen los datos devueltos por conceptnet y está formado por los siguientes campos:

  - @id: Describe como están conectados dos nodos(palabras), es decir, la relación que hay entre ellos Por ejemplo, como podemos ver en la figura 2.3, la relación entre gato y animal, es que el gato pertenece al grupo de los animales.

[5] http://api.conceptnet.io

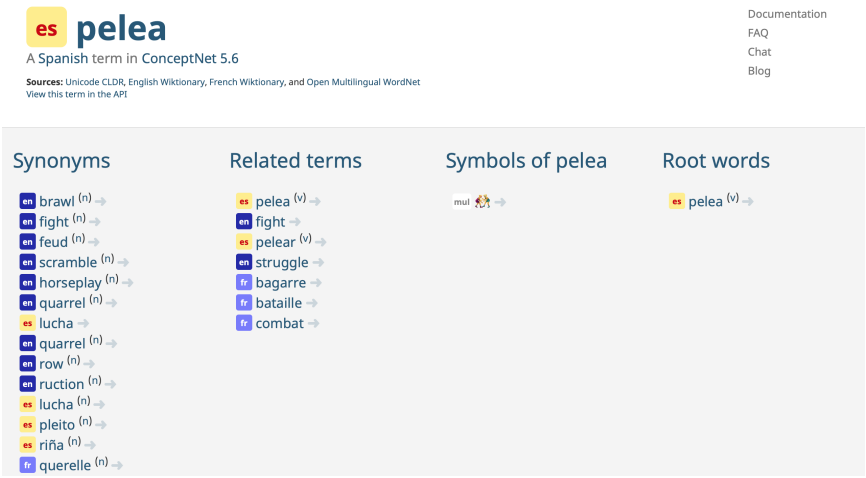

Figura 2.4: Resultados de búsqueda de una palabra

- @type:Describe el tipo del id, es decir, si es una relación (edge) o un término (nodo).
- dataset: No es muy importante, es un valor acerca de como está construido conceptnet.
- end: Describe el nodo destino, es decir, si hemos introducido la palabra "gatoün nodo destino sería por ejemplo .ªnimal", ya que es un término relacionado. Dentro de end hay varios campos ya explicados anteriormente, como @id o @type, sin embargo hay otros nuevos como "language", en el que se indica el idioma en el que esta la palabra destino y "term"que en este caso tiene el mismo valor que el @id
- license: Aporta información sobre como debe usarse la información proporcionada por conceptnet.
- rel: Describe la relación que hay entre la palabra origen y destino, dentro del cual hay tres campos: @id, @type y label, descritos anteriormente.
- sources: Indica por qué conceptnet guarda esa información, este campo como los anteriores, es un objeto que tiene su propio id, además de un campo .ªctivityïndicando la tarea del contribuidor que ha añadido esta palabra o si ha sido añadida mediante n proceso automático. Por último hay un campo çontributor", en el que aparece el nombre de la persona que ha añadido esta palabra.
- start: Describe el nodo origen, es decir, la palabra que hemos introducido en conceptnet para que haga la consulta, este campo

esta compuesto por elementos ya descritos como son: @id, @type, label, language y term.

- surfaceText: Indica de que frase del lenguaje natural se han extraido los datos que estan guardados en conceptnet
- weight: Indica la fiabilidad de la información guardada en conceptnet, siendo normal que su valor sea 1.0. Dicho valor es mayor cuanto más filedigna sean los datos.

- 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, "nextPage"que tiene un enlace a la siguiente página de la lista

[6]

Por ejemplo, si introducimos la palabra "dinero", especificando que el idioma es el español, ConceptNet te devuelve como sinónimos: lana, pasta, billete, plata. y como términos relacionados: dineral y moneda.

Dispone de una API (http://api.conceptnet.io/) que devuelve los datos en formato JSON, que utilizaremos para obtener los términos relacionados y sinónimos que necesitamos.

2.2.2. Alternativas a conceptnet

Una de las alternativas que hemos encontrado a conceptnet es Thesaurus *https://www.thesaurus.com* que es una aplicación web en la que introduces una palabra y obtienes términos relacionados y sinónimos en varios idiomas, principalmente en inglés, también te dice que tipo de palabra es (sustantivo, verbo... etc), esta opción en nuestro caso es inviable ya que la base de datos de palabras en castellano es bastante pobre.

Otra página relacionada que hemos encontrado ha sido Metaphor Magnet, que es otra aplicación web en la que al introducir una palabra, genera metáforas o palabras relacionadas, este servicio, unicamente esta implementado en inglés.

2.3. Figuras retóricas

Las figuras literarias (o retóricas) son formas no convencionales de utilizar las palabras, de manera que, aunque se emplean con sus acepciones habituales, se acompañan de algunas particularidades fónicas, gramaticales o semánticas, que las alejan de ese uso habitual, por lo que terminan por resultar especialmente expresivas. La metáfora, el símil y la analogía se basan en la comparación de dos conceptos: El de origen, que es el término literal (al

[6] http://conceptnet.io/

que la metáfora se refiere) se llama tenor y el de destino, que es el término figurado,el llamado vehículo. 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. (*Introducción al análisis retórico: tropos, figuras y sintaxis del estilo*) (**?**)

En este trabajo vamos a trabajar con tres tipos de figuras retóricas:

- Metáfora: Se refiere a una cosa mencionando otra, utiliza el desplazamiento de características similares entre dos conceptos con fines estéticos o retóricos. Por ejemplo, cuando una persona tiene muy buena memoria, se dice que tiene memoria de elefante. Ya que una de las características de los elefantes es que tienen buena memoria.

- Símil: Realiza una comparación entre dos términos. A pesar de que los símiles y las metáforas son similares, los símiles utilizan explícitamente conectores (por ejemplo, como, cual, que, o verbos). Por ejemplo, cuando nos referimos a una persona que es muy corpulenta, se dice que es como un oso, ya que los osos son muy grandes.

- Analogía: Es la comparación entre varios conceptos, indicando las características que permiten dicha relación. En la retórica, una analogía es una comparación textual que resalta alguna de las similitudes semánticas entre los conceptos protagonistas de dicha comparación. Por ejemplo,*sus ojos son azules como el mar.*

2.4. 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 en el cual estén programadas o ejecutadas en cualquier tipo de plataforma.(**?**) Según el W3C(*World Wide Web Consortium*)[7], 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: (**?**)

- Debe poder ser 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.

[7] https://www.w3.org/

- Debe contener 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 poder ser localizado. Deberemos tener algún mecanismo que nos permita encontrar un Servicio Web que realice una determinada función. De esta forma tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente el usuario.

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

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

2.4.1. Arquitectura Servicios Web

Hay que distinguir tres partes fundamentales en los servicios web:

- El proveedor: Es la aplicación que implementa el servicio y lo hace accesible desde Internet.
- El solicitante: Cualquier clienteque 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:

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

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

2.4.2. Ventajas de los Servicios Web

Las principales ventajas del uso de los servicios web son las siguientes:

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

2.4.3. Desventajas de los Servicios Web

El uso de servicios web también tiene algunas desventajas:

- 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.

```
{
  "@context": [
    "http://api.conceptnet.io/ld/conceptnet5.6/context.ld.json"
  ],
  "@id": "/c/es/pelea",
  "edges": [
    {
      "@id": "/a/[/r/RelatedTo/,/c/es/pelear/v/,/c/es/pelea/]",
      "@type": "Edge",
      "dataset": "/d/wiktionary/en",
      "end": {
        "@id": "/c/es/pelea",
        "@type": "Node",
        "label": "pelea",
        "language": "es",
        "term": "/c/es/pelea"
      },
      "license": "cc:by-sa/4.0",
      "rel": {
        "@id": "/r/RelatedTo",
        "@type": "Relation",
        "label": "RelatedTo"
      },
      "sources": [
        {
          "@id": "/and/[/s/process/wikiparsec/1/,/s/resource/wiktionary/en/]",
          "@type": "Source",
          "contributor": "/s/resource/wiktionary/en",
          "process": "/s/process/wikiparsec/1"
        },
        {
          "@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/pelear/v",
        "@type": "Node",
        "label": "pelear",
        "language": "es",
        "sense_label": "v",
        "term": "/c/es/pelear"
      },
      "surfaceText": null,
      "weight": 2.0
    },
    {
      "@id": "/a/[/r/Synonym/,/c/es/pelea/n/,/c/en/fight/n/]",
      "@type": "Edge",
      "dataset": "/d/wordnet/3.1",
      "end": {
```


Figura 2.5: Vista de JSON

```
{
  "@context": [
    "http://api.conceptnet.io/ld/conceptnet5.6/context.ld.json"
  ],
  "@id": "/c/es/pelea",
  "edges": [
    {
      "@id": "/a/[/r/RelatedTo/,/c/es/pelear/v/,/c/es/pelea/]",
      "@type": "Edge",
      "dataset": "/d/wiktionary/en",
      "end": {
        "@id": "/c/es/pelea",
        "@type": "Node",
        "label": "pelea",
        "language": "es",
        "term": "/c/es/pelea"
      },
      "license": "cc:by-sa/4.0",
      "rel": {
        "@id": "/r/RelatedTo",
        "@type": "Relation",
        "label": "RelatedTo"
      },
      "sources": [
        {
          "@id": "/and/[/s/process/wikiparsec/1/,/s/resource/wiktionary/en/]",
          "@type": "Source",
          "contributor": "/s/resource/wiktionary/en",
          "process": "/s/process/wikiparsec/1"
        },
        {
          "@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/pelear/v",
        "@type": "Node",
        "label": "pelear",
        "language": "es",
        "sense_label": "v",
        "term": "/c/es/pelear"
      },
      "surfaceText": null,
      "weight": 2.0
    },
    {
      "@id": "/a/[/r/Synonym/,/c/es/pelea/n/,/c/en/fight/n/]",
      "@type": "Edge",
      "dataset": "/d/wordnet/3.1",
      "end": {
```


Figura 2.6: Vista de JSON

Capítulo 3

Trabajo Realizado

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

3.1. Trabajo realizado por Irene

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

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

3.2. Trabajo realizado por Pablo

Al igual que mi compañera, lo primero que hicimos fue investigar como podíamos etiquetar las palabras, encontramos la librería ntlk 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 ntlk 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 servicio web prototipo y nos decantamos por utilizar Django como framework integrado en Pycharm, que es el entorno de desarrollo.

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

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

- Consistencia: La funcionalidad debe ser similar a otras aplicaciones que el usuario está acostumbrado a utilizar. En cuanto a la interfaz debe tener los mismos colores, iconos, formas, botones, mensajes de aviso... Por ejemplo, si el usuario está acostumbrado a que el botón de eliminar o cancelar sea rojo, no debemos añadirle uno de color verde.

- Usabilidad Universal: Debemos tener en cuenta las necesidades de los distintos tipos de usuario, como por ejemplo, atajos de teclado para un usuario experto o filtros de color para usuarios con deficiencias visuales.

- Retroalimentación activa: Por cada acción debe existir una retroalimentación legible y razonable por parte de la aplicación. Por ejemplo, si el usuario quiere guardar los datos obtenidos de la búsqueda, la aplicación debe informarle de si han sido guardados o no.

- Diálogos para conducir la finalización: El usuario debe saber en que paso se encuentra en cada momento. Por ejemplo, en un proceso de compra que conlleva varios pasos hasta la finalización de la misma, se le debe informar donde se encuentra y cuánto le queda para terminar.

- Prevención de errores: La interfaz debe ayudar al usuario a no cometer errores serios, y en caso de cometerlos se le debe dar una solución lo más clara y sencilla posible. Por ejemplo, deshabilitando opciones o indicando en un formulario el campo en el cual se ha producido el error sin perder la información ya escrita.

- Deshacer acciones fácilmente: Se debe dar al usuario la capacidad de poder deshacer o revertir acciones de una manera sencilla.

- Sensación de control: Hay que dar al usuario la sensación de que tiene en todo momento el control de la aplicación, añadiendo contenidos fáciles de encontrar y de esta forma no causarle ansiedad o frustración por utilizar nuestra aplicación.

- Reducir la carga de memoria a corto plazo: La interfaz debe ser lo más sencilla posible y con una jerarquía de información evidente, es decir, hay que minimizar la cantidad de elementos a memorizar por el usuario.

Capítulo 4

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Chapter 4

Conclusions and Future Work

Conclusions and future lines of work.

Apéndice A

Título

Contenido del apéndice

Apéndice B

Título

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

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

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

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