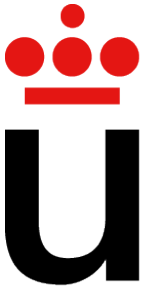

Universidad
Rey Juan Carlos

GRADO EN INGENIERIA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2020/2021

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN LEARNINGML

Autor : Isaac Merchán Blanco

Tutor : Dr. Gregorio Robles

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN LEARNINGML

Autor : Isaac Merchán Blanco

Tutor : Dr. Gregorio Robles

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2021, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2021

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Quiero agradecer a mis padres quienes han sido capaces de aguantar mis peores momentos durante esta carrera. A mi madre que desde pequeño siempre ha estado ayudandome con todo lo que he necesitado y confiando en mi. Y a mi padre que aunque no pase tanto tiempo con él me ha inculcado unos valores y una forma de ver la vida que me han llevado a ser quien soy a día de hoy.

Quería agradecer también a mis amigos por estar siempre dispuestos a sacar un rato para vernos y hablar, esos momentos de descanso también forman parte de esto y sin ellos nada sería lo mismo.

Gracias también a mis companeros con los que he pasado estos últimos años de mi vida y una etapa muy importante, sin ellos esto hubiera sido aún más difícil. Han hecho que las clases, las tardes de estudio, las prácticas y, en general, la carrera sea más divertida.

Por último, agreder a Gregorio su manera de explicar y su disposición para ayudarme siempre que he necesitado algo.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

Hoy en día vivimos en un mundo digital en el cual hay una cantidad inmensa de datos. La obtención de estos crece a un ritmo exponencial, solo en 2018 se generaron 33 zettabytes (un zettabyte equivale a 1.000 millones de terabytes) lo que equivale a 16.5 veces más que en 2009 [referencia web]. A pesar de esto, solo somos capaces de procesar el 0.5 % [referencia libro] y ese porcentaje cada vez va a menos ya que no se incrementa de la misma forma la capacidad de obtención que de procesado. El principal problema de esto está en la complejidad y cantidad de los datos pues los humanos no somos capaces muchas veces de extraer información útil de esos datos. Esto ha hecho que se fomente el estudio de distintos campos, uno de ellos es el *machine learning*. Estas limitaciones mencionadas anteriormente desaparecen en el momento en el cual el *machine learning* entre en juego, ya que es capaz de procesar grandes cantidades de datos complejos y transformarlos en información legible y fácil de analizar por los los humanos.

El *machine learning* (o aprendizaje automático) es la rama de la inteligencia artificial que trata de implementar el aprendizaje automático de máquinas a través de un entrenamiento. A pesar de ser un concepto relativamente nuevo, se basa en campos de estudio anteriores como la modelación estadística y o el reconocimiento de patrones.

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes[1].

[1] `http://www.tallerdeescritores.com/errores-de-escritura-frecuentes`

1.1. Presentación de la aplicación

Este Trabajo Fin de Grado tiene como principal objetivo la integración de los algortimos de aprendizaje automático de Naive Bayes y KNN en la web de LearningML que actualmente solo utiliza el método de redes neuronales para clasficar imágenes o texto.

LearnignML es una web creada con Angular para que cualquier persona sin grandes conocimientos sobre el aprendizaje autmático pueda crear un modelo, entrenarlo y después probarlo. La web además permite descargar el modelo creado en un archivo JSON para después poder cargarlo sin necesidad de crearlo de nuevo, esto es muy útil ya que de esta manera no se tienen que volver a introducir todas las entradas para el aprendizaje, si no que directamente se puede importar un modelo creado previamente e introducir una entrada para ver su funcionamiento. También tiene la opcion de registrarse y poder guardar los modelos en tu cuenta o incluso tener proyectos compartidos entre varias personas. Por último, permite crear un modelo en Scratch y utilizarlo en la web aunque esta parte no se verá durante este trabajo.

Por cierto, a veces me comentáis que no os compila por las tildes. Eso es un problema de codificación. Al guardar el archivo, guardad la codificación de "ISO-Latin-1" a "UTF-8" (o viceversa) y funcionará.

1.1.1. Estilo

Recomiendo leer los consejos prácticos sobre escribir documentos científicos en LaTeX de Diomidis Spinellis[2].

Lee sobre el uso de las comas[3]. Las comas en español no se ponen al tuntún. Y nunca, nunca entre el sujeto y el predicado (p.ej. en "Yo, hago el TFG" sobre la coma).

A continuación, viene una figura, la Figura 1.1. Observarás que el texto dentro de la referencia es el identificador de la figura (que se corresponden con el "label" dentro de la misma). También habrás tomado nota de cómo se ponen las "comillas dobles" para que se muestren correctamente. Nota que hay unas comillas de inicio (``) y otras de cierre ('') , y que son diferentes. Volviendo a las referencias, nota que al compilar, la primera vez se crea un diccionario con

[2] https://github.com/dspinellis/latex-advice

[3] http://narrativabreve.com/2015/02/opiniones-de-un-corrector-de-estilo-11-recetas-par
html

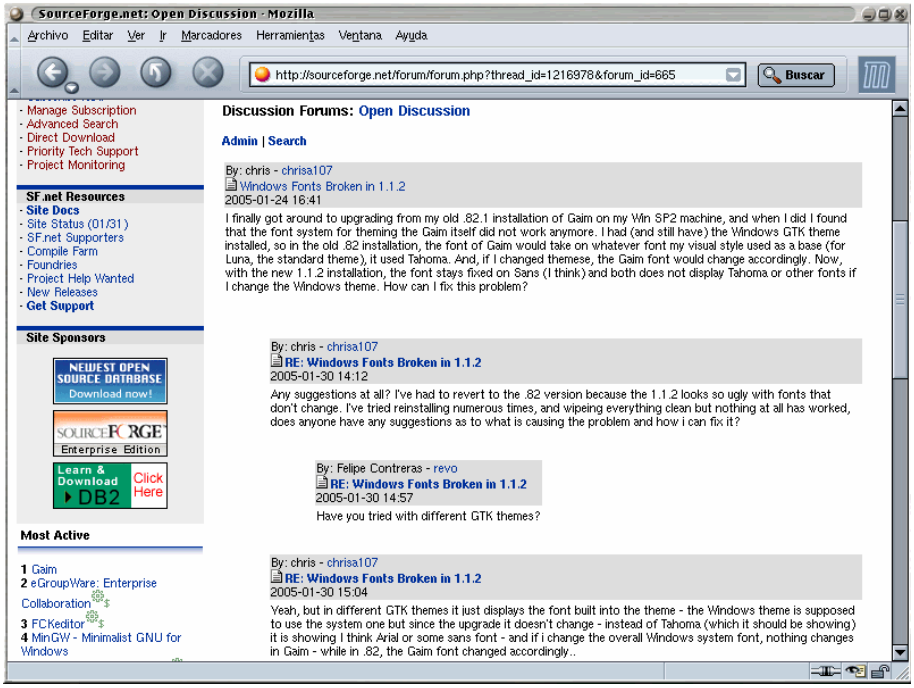

Figura 1.1: Página con enlaces a hilos

las referencias, y en la segunda compilación se “rellenan” estas referencias. Por eso hay que compilar dos veces tu memoria. Si no, no se crearán las referencias.

A continuación un bloque “verbatim”, que se utiliza para mostrar texto tal cual. Se puede utilizar para ofrecer el contenido de correos electrónicos, código, entre otras cosas.

```
From gaurav at gold-solutions.co.uk  Fri Jan 14 14:51:11 2005
From: gaurav at gold-solutions.co.uk (gaurav_gold)
Date: Fri Jan 14 19:25:51 2005
Subject: [Mailman-Users] mailman issues
Message-ID: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>
```

1.2. Estructura de la memoria

La memoria empieza con un resumen, en el que se explica brevemente en que consiste el proyecto, así como las tecnologias que se han usado para llevarlo a cabo. A continuación, se sitúan los agradecimiento y los índices, tanto el general como el de figuras. Finalmente, van los 6 capítulos en lo que se divide la memoria.

- **Capítulo 1: Introducción.** En el primer capítulo se hace una breve explicación de en que

consiste la web de LearningML y la estructura que va a seguir el trabajo.

- **Capítulo 2: Objetivos.** En este capítulo se describen los objetivos principales y específicos del trabajo y la planificación temporal de este.
- **Capítulo 3: Estado del arte.** Se trata de una breve explicación sobre la mayoría de las teconologías utilizadas en el desarrollo de la aplicación web.
- **Capítulo 4: Diseño e implementación.** Se desccribe de manera detallada tanto los cambios previos en la estructura como la arquitectura final de la web así como la implementación de los dos modelos de aprendizaje.
- **Capítulo 5: Experimentos y validación.** Se comprueba si funcionan correctamente las funcionalidades implementadas en la web de LearningML.
- **Capítulo 6: Conclusiones.** Por último, se repasa todo lo aprendido durante el desarrollo del proyecto y se comentan algunas mejoras o funciones que se podrían implementar en un futuro.

Capítulo 2

Objetivos

2.1. Objetivo general

Aquí vendría el objetivo general en una frase: Mi trabajo fin de grado consiste en crear de una herramienta de análisis de los comentarios jocosos en repositorios de software libre alojados en la plataforma GitHub. Mi trabajo fin de grado consiste en añadir los algortimos de aprendizaje automático de Bayes y KNN de manera intuitiva e independientemente de si se quieren clasificar textos o imágenes a la web de LearningML. Recuerda que los objetivos siempre vienen en infinitivo.

2.2. Objetivos específicos

Para la realización del objetivo general se han planteado los siguientes objetivos específicos:

- Transformar las imagenes o texto de tal forma que obtengamos una misma estructura que se pueda pasar como entrada a los distintos algoritmos.
- Implementar el algortimo de Bayes y KNN para que funcionen como dos clasificadores de tal forma que primero entrenen con las entradas de aprendizaje y luego evaluen otra entrada proporcionada por el usuario y la asigne correctamente a uno de los grupos creados previamente.
- Añadir en la interfaz una manera de poder elegir el tipo de algoritmo que se quiere utilizar de forma independiente a si los elementos a clasificar son texto o imágenes.

- Facilitar su uso para que sea sencilla y que todas las personas que la usen puedan entender y comprender su funcionamiento.

Los objetivos específicos se pueden entender como las tareas en las que se ha desglosado el objetivo general. Y, sí, también vienen en infinitivo.

2.3. Planificación temporal

La primera idea que tuve de empezar con el Trabajo Fin de Grado (TFG) fue en Enero de 2021 y quería hacerlo a la vez que las prácticas entonces me puse en contacto con Gregorio y me comentó esto. Me llamo la atención la web de LearningML ya que recuerdo haber utilizado Scratch en el colegio y me parecía interesante aportar algo aunque sea de manera indirecta a que los niños se interesen por la programación. En principio este proyecto iba a ser tanto prácticas como TFG y le iba a poder dedicar bastante horas todos los dias, sin embargo, me aceptaron en unas prácticas y no tuve tanto tiempo ya que estas eran de jornada completa. Además de esto, aun tenía una asignatura pendiente asi que era difícil establecer un horario concreto para dedicarle tiempo al proyecto. Cuando no tenía un examen cerca, al salir de trabajar me ponía con ello un rato y luego los fines de semana por la mañana también le dedicaba algo de tiempo. Recuerdo que primero tuve que aprender como funcionaba Angular y Typescript con un curso que tiene Juanda. Practicando y familiarizandome con la forma de trabajar en Angular. Después tuve varias reuniones con Juanda (Juan David Rodríguez García, creador de LearningML, esto irá en la introduccion) para ver como funcionaba la aplicacion de LearningML y donde había que añadir los cambios. También tuve que buscar que librerias ya construidas de Bayes y KNN podiamos adaptar para usarlas en la web y finalmente ponerme a ello. Por último, quedaba la tarea de trasladar todo eso al papel. Me puse en contacto con mi tutor y enseguida me paso una plantilla con los apartados a rellenar y...continuará

Capítulo 3

Estado del arte

En este apartado veremos tanto las tecnologias utilizadas como una breve explicación sobre lo que es el modelo de Bayes y de KNN.

3.1. HTML5

3.2. TypeScript

3.3. Node.js

3.4. JSON

3.5. Angular

Angular [2] es un *framework* de código abierto diseñado por Google. Permite crear aplicaciones web de una sola página, lo que se denomina SPA (Single Page Application). Utiliza TypeScript y HTML para el desarrollo de las aplicaciones web.

Angular permite separar el *frontend* del *backend* y sigue un modelo MVC (Modelo-Vista-Controlador) el cual se verá como funciona en el siguiente apartado. De esta forma, las modificaciones y las actualizaciones de las aplicaciones son rápidas y sencillas.

Una de las principales ventajas de este *framework* y de las páginas SPA es la velocidad de

carga entre las diferentes vistas de la aplicación. Cuando se cambia de vista no se recarga la página si no que éstas se cargan de manera dinámica, rápida y reactiva.

Angular dispone de varias versiones, aportando todas ellas mejoras en dicho *framework*. La primera versión de Angular, se conoce como AngularJS, y es la más distinta a las demás. El resto son actualizaciones de Angular 2. Cuando se definió la segunda versión se decidió modificar el nombre del *framework* y dejarlo como Angular. La última versión estable es la versión 10, utilizada por LearningML.

La arquitectura de una aplicación Angular se basa en clases de 4 tipos distintos (módulos, componentes, servicios y directivas). Estas clases se identifican a través de decoradores, los cuales permiten cargar los metadatos necesarios que determinan a Angular como debe usar dichas clases.

Módulos.

Los módulos suministran el contexto de compilación de los componentes, es decir, es aquí donde se definen los diferentes componentes que van a formar la aplicación, las dependencias, las clases que actúan como servicios en la aplicación y las rutas de navegación que establecen las vistas de la aplicación.

Toda aplicación angular tiene un módulo principal o raíz. Este módulo suele recibir el nombre de *AppModule* y es el que inicia el sistema de arranque de la aplicación.

Al igual que en JavaScript, los módulos de Angular permiten importar y exportar funcionalidades proporcionadas por otros módulos, por lo que una aplicación puede tener más de un módulo, siendo cada uno de ellos independiente. La manera en que se organizan estos módulos ayuda a la creción de aplicaciones más complejas y a la reutilización de código. Además, permite que la carga inicial sea mínima, ya que cada módulo se carga a petición, es decir, cuando se va a utilizar.

Componentes.

Las aplicaciones Angular tienen un componente que es nexo de unión entre los diferentes componentes que forman la aplicación y el modelo de objeto del documento de la página (DOM). Los componentes son los que tienen la lógica y los datos de la aplicación. Son los que controlan las diferentes plantillas HTML que se cargan cuando se modifican las URLs dentro de la aplicación. Estas plantillas HTML son las vistas que forman la interfaz de usuario.

De la misma forma que las aplicaciones diseñadas con Angular pueden tener más de un

módulo, también pueden tener subcomponentes, que se pueden relacionar entre sí mediante dos tipos de vinculación de datos, eventos y propiedades. A través de los eventos, la aplicación responde a las entradas del usuario actualizando los datos. A través de las propiedades, se envían valores calculados de los datos de la aplicación a los HTMLs que forman las vistas. Esta vinculación de datos se puede producir en ambas direcciones, es decir, igual que los datos de la aplicación pueden modificar los HTMLs, los cambios en el DOM pueden modificar los datos de la aplicación.

El decorador @Component es el que determina que una clase actúe como un componente.

Servicios

Los servicios son clases en las que se definen datos o funcionalidades generales que no están asociadas a una determinada vista. Son usados para compartir datos y operaciones entre componentes. También es donde se suele realizar toda la operativa de las peticiones a la API de las aplicaciones. Deben llevar el decorador @Injectable, pues es el que permite obtener los metadatos necesarios para que otras clases puedan inyectar sus dependencias.

Directivas

Las directivas son clases en las que se definen los términos claves que se usan en las plantillas. Cuando se carga una vista, Angular procesa estos términos clave que modifican el HTML y el DOM de la aplicación. Por tanto, una plantilla, además de utilizar HTML para definir la vista, usa marcas propias de Angular que son estas directivas.

Existen dos clases de directivas: (i) las de atributo, que modifican el comportamiento del componente, y (ii) las estructurales, que únicamente modifican la apariencia.

Angular dispone de un enrutador, módulo que contiene un servicio para definir las rutas de navegación de la aplicación. Este módulo se ajusta a las convenciones de los navegadores. Es decir, tanto si se pone la URL en la barra de direcciones, como si se pincha un enlace en la aplicación, como si se hace *click* en el botón de retroceso o avance, se carga a la página correspondiente. El enrutador es el que muestra u oculta una vista. Cada vez que se modifica la URL, el enrutador se encarga de añadir la ruta en el historial del navegador, es esto lo que permiete que los botones de retroceso y avance funcionen. Cada ruta de navegación está asociado a un componente.

3.5.1. MVC

Las aplicaciones web se suelen desarrollar siguiendo una serie de pautas. El patrón que sigue Angular es el Modelo Vista Controlador (MVC), cuyos elementos son [1] :

- **Modelo:** es la representacion de los datos de toda la aplicación y, por tanto, maneja las consultas y todas las acciones relacionadas con base de datos en caso de que las hubiera. También tratará los datos para cumplir con las funcionalidades de la aplicacióon.
- **Vista:** recibe los datos y los sirve al usuario final en la interfaz. Ni el controlador ni el modelo se preocupan por el diseño ni la apariencia final, de todo el apartado visual eso se encargarán las vistas.
- **Controlador:** es el elemento principal, encargado de responder a las acciones de entrada y lanzar una serie de funciones asignadas previamente. Es decir, recibe un evento de entrada, se comunica con el modelo en caso de que sea necesario, consultan las vistas y responden al evento con una salida.

Las principales ventajas del MVC son las siguientes:

- Mejora la escalabilidad, puesto que es un modelo sencillo en el que cada tipo de lógica esta separado.
- Facilita mantenimiento.
- Ofrece la posibilidad de reutilizar los componentes.

3.5.2. Angular CLI

Angular CLI (Command Line Interface) [3] es una herramienta de línea de instrucciones creada por el equipo de Angular. Es muy util a la hora de iniciar una aplicación web diseña con Angular, ya que con una sola instrucción, se genera el esqueleto de carpetas y archivos necesarios de la aplicación. Además, contiene herramientas predefinidas que ayudan al desarrollo y mantenimiento de este tipo de aplicaciones. Al crear una aplicación web con Angular CLI, dentro de la estructura de archivos, se crea un archivo de configuración, en el cual se añaden las

dependencias necesarias para que la aplicación web compile y ejecute. Este archivo se va modificando conforme se van creando los componentes, servicios o directivas. Para ello, Angular CLI tiene instrucciones que permiten crear estas clases de manera sencilla, creando los distintos archivos que conforman un componente, un servicio o una directiva. Entre las herramientas predefinidas destaca el compilador, el sistema de testing y el servidor web.

Puedes citar libros, como el de Bonabeau et al., sobre procesos estigmérgicos [4]. Nota que el ~ añade un espacio en blanco, pero no deja que exista un salto de línea. Imprescindible ponerlo para las citas.

Citar es importantísimo en textos científico-técnicos. ¿Dónde puedo encontrar textos científicos que referenciar? Un buen sitio es Google Scholar[1]. Por ejemplo, si buscas por "stigmergy libre software" para encontrar trabajo sobre software libre y el concepto de *estigmergia* (¿te he comentado que me gusta el concepto de estigmergia ya?), encontrarás un artículo que escribí hace tiempo cuyo título es "Self-organized development in libre software: a model based on the stigmergy concept". Si pulsas sobre las comillas dobles (entre la estrella y el "citado por ...", justo debajo del extracto del resumen del artículo, te saldrá una ventana emergente con cómo citar. Abajo a la derecha, aparece un enlace BibTeX. Púlsalo y encontrarás la referencia en formato BibTeX, tal que así:

```
@inproceedings{robles2005self,
  title={Self-organized development in libre software:
         a model based on the stigmergy concept},
  author={Robles, Gregorio and Merelo, Juan Juli\'an
          and Gonz\'alez-Barahona, Jes\'us M.},
  booktitle={ProSim'05},
  year={2005}
}
```

Copia el texto en BibTeX y pégalo en el fichero `memoria.bib`, que es donde están las referencias bibliográficas. Para incluir la referencia en el texto de la memoria, deberás citarlo, como hemos hecho antes con [4], lo que pasa es que en vez de el identificador de la cita anterior (bonabeau:_swarm), tendrás que poner el nuevo (robles2005self). Compila el fichero `memoria.tex` (`pdflatex memoria.tex`), añade la bibliografía (`bibtex memoria.aux`) y vuelve a

[1] `http://scholar.google.com`

Uno	2	3
Cuatro	5	6
Siete	8	9

Cuadro 3.1: Ejemplo de tabla. Aquí viene una pequeña descripción (el *caption*) del contenido de la tabla. Si la tabla no es autoexplicativa, siempre viene bien aclararla aquí.

compilar `memoria.tex` (`pdflatex memoria.tex`)...y *voilà* ¡tenemos una nueva cita [5]!

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página del GSyC[2].

3.6. Modelo de Naive Bayes

3.7. Modelo KNN o K vecinos más cercanos

Hemos hablado de cómo incluir figuras. Aquí un ejemplo de tabla, la Tabla 3.7 (siento ser pesado, pero nota cómo he puesto la referencia).

[2] `http://gsyc.es`

Capítulo 4

Diseño e implementación

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

4.1. Reorganizar LearningML

4.2. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a "vista de pájaro").

Por ejemplo, puedes verlo en la figura 4.1. LaTeX pone las figuras donde mejor cuadran.

A veces queda un poco raro, pero es la filosofía de LaTeX: tú al contenido, que yo me encargo de la maquetación.

Recuerda que toda figura que añadas a tu memoria debe ser explicada. Sí, aunque te parezca evidente lo que se ve en la figura 4.1, la figura en sí solamente es un apoyo a tu texto. Así que explica lo que se ve en la figura, haciendo referencia a la misma tal y como ves aquí. Por ejemplo: En la figura 4.1 se puede ver que la estructura del *parser* básico, que consta de seis componentes diferentes: los datos se obtienen de la red, y según el tipo de dato, se pasará a un *parser* específico y bla, bla, bla...

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

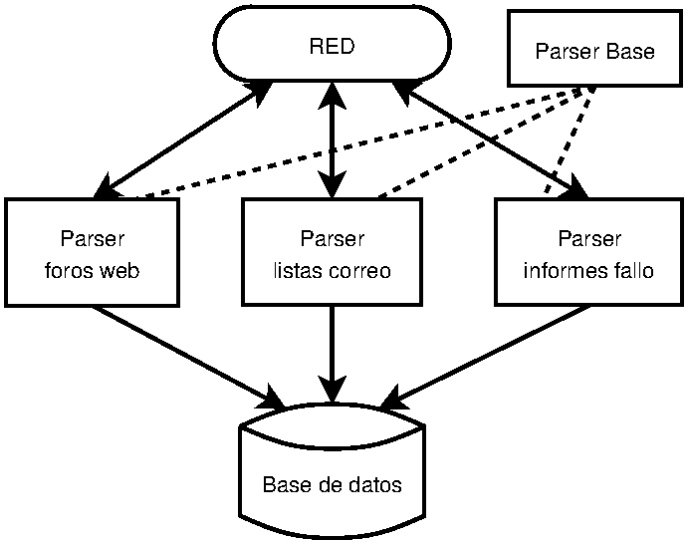

Figura 4.1: Estructura del parser básico.

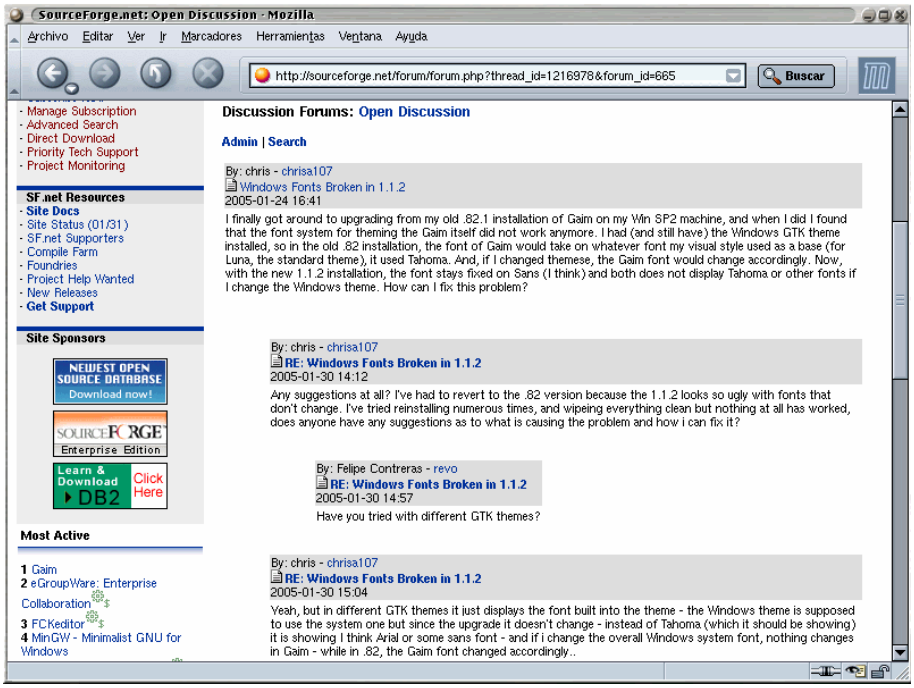

Figura 4.2: Página con enlaces a hilos

4.3. Crear el clasificador de Naive Bayes

4.4. Crear el clasificador de KNN

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

5.1. Prueba del clasificador de Naive Bayes

5.2. Prueba del clasificador de KNN

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

6.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

6.4. Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Apéndice A

Manual de usuario

Esto es un apéndice. Si has creado una aplicación, siempre viene bien tener un manual de usuario. Pues ponlo aquí.

Bibliografía

[1] Características y ventajas de la arquitectura mvc.
https://marketiweb.com/empresa/blog/item/114-que-es-la-arquitectura-mvc-y-cuales-son-sus-ventajas.

[2] Página de Angular.
https://angular.io.

[3] Página de Angular Cli.
https://cli.angular.io.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Articial Systems*. Oxford University Press, Inc., 1999.

[5] G. Robles, J. J. Merelo, and J. M. González-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *ProSim'05*, 2005.