TFG: Análisis Emocional para la Inclusión Digital

Gema Eugercios Suárez
Paloma Gutiérrez Merino
Elena Kaloyanova Popova

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Mayo 2018

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

TFG: Análisis Emocional para la Inclusión Digital

Informe técnico del departamento
Ingeniería del Software e Inteligencia Artificial
IT/2009/3

Versión 1.0

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Mayo 2018

ISBN 978-84-692-7109-4

Al duque de Béjar
y
a tí, lector carísimo

I can't go to a restaurant and order food because I keep looking at the fonts on the menu.
Donald Knuth

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, nosotros queremos agradecer al Word de Microsoft el habernos forzado a utilizar LaTeX. Cualquiera que haya intentado escribir un documento de más de 150 páginas con esta aplicación entenderá a qué nos referimos. Y lo decimos porque nuestra andadura con LaTeX comenzó, precisamente, después de escribir un documento de algo más de 200 páginas. Una vez terminado decidimos que nunca más pasaríamos por ahí. Y entonces caímos en LaTeX.

Es muy posible que hubíeramos llegado al mismo sitio de todas formas, ya que en el mundo académico a la hora de escribir artículos y contribuciones a congresos lo más extendido es LaTeX. Sin embargo, también es cierto que cuando intentas escribir un documento grande en LaTeX por tu cuenta y riesgo sin un enlace del tipo "*Author instructions*", se hace cuesta arriba, pues uno no sabe por donde empezar.

Y ahí es donde debemos agradecer tanto a Pablo Gervás como a Miguel Palomino su ayuda. El primero nos ofreció el código fuente de una programación docente que había hecho unos años atrás y que nos sirvió de inspiración (por ejemplo, el fichero `guionado.tex` de TeXiS tiene una estructura casi exacta a la suya e incluso puede que el nombre sea el mismo). El segundo nos dejó husmear en el código fuente de su propia tesis donde, además de otras cosas más interesantes pero menos curiosas, descubrimos que aún hay gente que escribe los acentos españoles con el `\'{\i}`.

No podemos tampoco olvidar a los numerosos autores de los libros y tutoriales de LaTeX que no sólo permiten descargar esos manuales sin coste adicional, sino que también dejan disponible el código fuente. Estamos pensando en Tobias Oetiker, Hubert Partl, Irene Hyna y Elisabeth Schlegl, autores del famoso "The Not So Short Introduction to LaTeX 2_ε" y en Tomás

Bautista, autor de la traducción al español. De ellos es, entre otras muchas cosas, el entorno `example` utilizado en algunos momentos en este manual.

También estamos en deuda con Joaquín Ataz López, autor del libro "Creación de ficheros LaTeX con GNU Emacs". Gracias a él dejamos de lado a WinEdt y a Kile, los editores que por entonces utilizábamos en entornos Windows y Linux respectivamente, y nos pasamos a emacs. El tiempo de escritura que nos ahorramos por no mover las manos del teclado para desplazar el cursor o por no tener que escribir `\emph` una y otra vez se lo debemos a él; nuestro ocio y vida social se lo agradecen.

Por último, gracias a toda esa gente creadora de manuales, tutoriales, documentación de paquetes o respuestas en foros que hemos utilizado y seguiremos utilizando en nuestro quehacer como usuarios de LaTeX. Sabéis un montón.

Y para terminar, a Donal Knuth, Leslie Lamport y todos los que hacen y han hecho posible que hoy puedas estar leyendo estas líneas.

Resumen

...

...

...

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

...

...

Resumen: Este capítulo sirve como introducción al trabajo que se va a realizar.

1.1. Introducción

...

Notas bibliográficas

Citamos algo para que aparezca en la bibliografía... (Bautista et al., 1998)

Y también ponemos el acrónimo CVS para que no cruja.

Ten en cuenta que si no quieres acrónimos (o no quieres que te falle la compilación en "release" mientras no tengas ninguno) basta con que no definas la constante `\acronimosEnRelease` (en `config.tex`).

En el próximo capítulo...

...

Capítulo 2

Estado del arte

...

...

Resumen: En este capítulo se van a tratar los aspectos más importantes tanto de la computación afectiva como de las diferentes tecnologías y metodologías que se van a utilizar. En primer lugar, en la sección 2.1, se define la computación afectiva y sus posibles aplicaciones, también se explican los distintos diccionarios afectivos ya existentes que permiten la marcación emocional de textos. En la sección 2.2 se introduce la tecnología que se va a utilizar para implementar el trabajo, los servicios web. En la sección 2.3 se fijan los conceptos relacionados con la metodología Scrum, la cual hemos seguido durante todo el TFG. En la sección 2.4 se explican las bases de la integración continua aplicada al desarrollo de software y cómo se va a aplicar en este trabajo.

2.1. Computación Afectiva

Todo el mundo sabe lo que son las emociones hasta que llega el momento de definirlas. Podríamos definir las emociones como reacciones automáticas que nuestro cuerpo experimenta ante un determinado estímulo. Sin embargo, todas ellas luego derivan en sentimientos más prolongados en el tiempo.

Existen 6 emociones básicas (enfado, miedo, asco, alegría, sorpresa y tristeza), las cuales podemos representar e interpretar gracias a los gestos y movimientos.

La computación afectiva (Picard,1997) es el estudio y el desarrollo de sistemas y dispositivos capaces de percibir, medir e interpretar las emociones humanas.

Esta rama de la computación permite un avance notable en la inteligencia artificial, hasta tal punto que los ordenadores lleguen a adaptarse a

los humanos, sus necesidades y estados de ánimo. Los seres humanos están rodeados de emociones, en cualquier ámbito de su vida, tanto de las suyas propias como las de las personas con las que se comunican. Tanta importancia tienen para nosotros que influyen no sólo en nuestra comunicación, sino también en nuestro aprendizaje y toma de decisiones. Por ello, resulta artificial y en ocasiones incluso frustrante intentar comunicarse con una máquina que no es capaz de expresar sentimientos. La computación afectiva pretende mejorar la interacción hombre-máquina haciéndola más natural y asequible.

El funcionamiento de este tipo de sistemas se basa en identificar el estado emocional del sujeto a través de diferentes fuentes (voz, expresiones, señales fisiológicas, palabras...) y procesar la información para clasificarla y aprender de ella. Clasificar la información de entrada puede resultar complicado ya que se suelen recibir varias señales diferentes a la vez, lo que hace necesario utilizar técnicas de priorización para determinar cuáles son las que más aportan a la hora de analizar y gestionar la información. Una vez identificada la emoción predominante, el sistema responde adecuandose a ella. La salida dependerá del tipo de sistema y las herramientas de las que dispone este para expresar su respuesta (colores, sonidos, emoticonos...) En casos más complejos aplicados a robótica o modelado, la salida producida es una simulación de la respuesta que produciría un ser humano ante los estímulos recibidos imitando su expresión corporal, voz o gestos faciales. Por ejemplo los robots Geminoid (IEEE Sprectrum, 08/04/2010)

La computación afectiva tiene multitud de aplicaciones, ya que como se ha mencionado antes, las emociones están presentes en todos los ámbitos de la vida de una persona. Por lo tanto, puede aplicarse a áreas muy diferentes entre sí como:

- **Marketing**: Actualmente una de sus aplicaciones más explotadas. Poder evaluar la reacción emocional de alguien ante un anuncio o producto es una estrategia comercial que ya está siendo utilizada cada vez por más empresas.

- **Salud**: Principalmente detección del estrés para minimizar sus efectos y aprender a controlarlo. Es posible inferir el nivel de estrés de una persona midiendo sus señales fisiológicas (ritmo cardiaco, respiración...) y si este nivel es demasiado alto se reaccionará en consecuencia según el tipo de sistema. Se puede aplicar de forma similar a las fobias.

- **Entretenimiento**: La industria de los videojuegos ha crecido mucho en los últimos años e introducir este tipo de tecnología permite a las compañías crear juegos más adaptables y cercanos al jugador, lo que atrae a más público y mejora la experiencia de juego (Ng, Khong, Thwaites, 2012).

- **Robótica**: El mayor problema de los robots diseñados para interactuar

con humanos es la carencia de emociones. Algunos de ellos llegan a producir una sensación de incomodidad. Dotar a este tipo de robots de cierta "humanidad"no sólo haría más cómodo el tratar con ellos sino que podrían realizar tareas como el acompañamiento de personas mayores. (Riek, Paul, and Robinson, 2010)

- **Accesibilidad**: El paradigma actual obliga a cualquier usuario a adaptarse a las máquinas sin tener en cuenta las dificultades particulares que pueda tener. Es complicado crear una máquina capaz de adaptarse a todas las circunstancias. El uso de computación afectiva permite facilitar la inversión del paradigma para que las máquinas utilicen las emociones del usuario para adaptarse a él.

En este trabajo vamos a centrarnos en el último área, la accesibilidad. En particular, en facilitar a personas que padecen Trastornos del Espectro Autista (TEA) el entendimiento de textos. Trataremos de analizar un texto de entrada para identificar las emociones predominantes y en que medida se presentan, evitando las posibles ambigüedades que pueda haber. Las emociones con las que trataremos son las básicas: alegría, tristeza, miedo, sorpresa y enfado. Una vez identificadas las emociones se etiquetarán para hacerlas más explícitas mediante emoticonos.

Para deducir la emoción que transmite una palabra específica utilizaremos un diccionario afectivo basado en otros ya existentes.

2.1.1. Métodos existentes para el marcado de texto emocional

Los métodos existentes para el marcado de texto emocional podrían clasificarse en cinco categorías básicas: keyword spotting, afinidad léxica, procesamiento estadístico del lenguaje natural, métodos basados en el conocimiento del mundo real y métodos manuales.

- **Keyword spotting**: El marcado del texto se basa en la aparición de palabras emocionales como happy, sad... Un ejemplo de este método es ANEW (Bradley y Lang, 1999), se trata de una base de datos de 3.109 palabras con su emoción asociada. Las principales desventajas de este método son: problemas cuando la negación está implicada en la frase y dependencia del método en aspectos superficiales cuando en la práctica existen muchas frases cuyo contenido emocional no se encuentra en los adjetivos.

- **Afinidad léxica**: Este método no solo detecta palabras obviamente emocionales como los adjetivos, sino que asigna al resto de palabras una afinidad con las distintas emociones. Este método tiene dos problemas fundamentales: emplea tan solo las palabras sin tener en cuenta el

contexto en el que se encuentran por lo que puede fallar con la aparición de las negaciones y ademas este tipo de métodos suelen entrenarse con un corpus lo que dificulta el desarrollo de un modelo reutilizable e independiente del dominio.

- **Procesamiento estadístico**: Consiste en alimentar a un algoritmo de aprendizaje con un amplio corpus de textos marcados emocionalmente. Este método ha sido empleado en el proyecto Webmind (Goertzel et al., 2000).
- **Métodos manuales**: Estos métodos implican el modelado de distintos estado emocionales en términos de modelos afectivos basados en teorías psicológicas sobre las necesidades, los deseos y las metas de los seres humanos. Un ejemplo es el DAYDREAMER (Dyer, 1987)
- **Métodos basados en el conocimiento del mundo real**: Este método no solo mira los aspectos superficiales del texto, sino que va más allá, evalúa la calidad afectiva de la semántica subyacente que contiene el texto. Un ejemplo es el marcador llevado a cabo por Liu, Lieberman y Selker (Liu, Lieberman, y Selker, 2002)

Las técnicas basadas en el procesamiento estadístico del lenguaje natural solo funcionan con un texto de entrada lo suficientemente extenso, los métodos manuales precisan un amplio análisis y entendimiento de los textos y hacen muy difícil la generalización y los métodos basados en el conocimiento del "mundo real"necesitan un amplio conocimiento denominado "mundo real". (Francisco, Gervás, 2006)

2.1.2. Diccionarios Afectivos Existentes

Un diccionario afectivo es un diccionario en el que las palabras se encuentran marcadas con etiquetas afectivas. Algunos ejemplos de etiquetas afectivas serían: alegría, tristeza, sorpresa, miedo...

En las siguientes secciones presentamos los diccionarios afectivos mas representativos.

2.1.2.1. LASSWELL VALUE DICTIONARY Y GENERAL INQUIRER

Fue a comienzos de los años 60 cuando Stone y Lasswell, ambos investigadores, comenzaron a construir diccionarios en lo que las palabras estaban marcadas con etiquetas afectivas. El diccionario Lasswell clasifica sus palabras de manera binaria y en ocho categorías básicas (riqueza, poder, respeto, rectitud, habilidad, iluminación, afecto y bienestar). Además de estas, el esquema de este diccionario distingue entre objetivos sustantivos y los elementos y atributos del proceso de distribución de valores, es decir, evaluación

personal y asignación social. La clasificación social del contenido mantiene estas distinciones. Las preocupaciones con valores particulares se clasifican en objetivos sustantivos particulares. El esquema de clasificación también distingue varios tipos de transacciones de valor.

2.1.2.2. DICCIONARIO DE HATZIVASSILOGLOU Y MCKEOWN

Este diccionario fue creado en 1997, y tarta de encontrar etiquetas como positive o negative de manera automatizada a través del análisis de un corpus. Hatzuvassiloglou y McKeown tomaron una serie de adjetivos que aparecían de manera frecuente y decidieron darles una orientación, empleando un análisis estadístico en el que cuando dos adjetivos aparecían juntos, siguiendo el patrón adjetivo1 y adjetivo 2, devolvía una orientación positiva o negativa. Así, pudieron obtener un diccionario de adjetivos clasificados en positivos y negativos.

2.1.2.3. WORDNET AFFECT

Se trata de una extensión de WordNet Domains, que incluye un subconjunto de synsets adecuados para representar conceptos afectivos correlacionados con palabras afectivas. De manera similar a nuestro método para etiquetas de dominio, asignamos a una serie de sintonias de WordNet una o mas etiquetas afectivas. En particular, los recursos afectivos que representan el estado emocional son individualizados por sintonizadores marcados por la emocion de la etiqueta A. Tambien existen otras etiquetas para aquellos conceptos que representan estados de animo, situaciones que provocan emociones o respuestas emocionales. El recurso se amplió con un conjutno de etiquetas A adicionales (llamadas categorias emocionales), organizadas jerarquicamente, con el fin de especializar sysnsets con una emocion de etiqueta. La estructura jerarquica de las nuevas etiquetas a se modelo en la relaciones de hiperonimos de wornet. En una segunda etapa, se definen cuatro etiquetas a-adicionales: positivo, negativo, ambiguo, neutral. Otra propiedad importante para el lexico afectico en lo que se se refiere principalmente a la interpretacion adejtivas es la dimension estacionaria/causativa.

2.1.2.4. WHISSELL'S DICTIONARY OF AFFECTA IN LANGUAGE (DAL)

Este diccionario es un recurso diseñado con el fin de medir el significado emocional de las palabras y textos. El objetivo del diccionario es comparar las palabras con una lista de 8742 palabras que han sido marcadas por distints personas en terminos de las dimensiones activacion y evaluacion e imagenes.

2.1.2.5. DICCIONARIO DE TURNEY Y LITTMAN

Turney y Littman encontraron una manera mas eficiente para decidir cuando una palabra se podía considerar positiva o negativa. Dado un conjunto de palabras, que previamente ellos conocían como positivas o negativas, probaron como de frecuentemente podía aparecer una palabra en un contexto con un conjunto de palabras positivas o negaticas. Empleando la informacion de estudios cercanos y estadísticas clasificaron como positivas todas aquellas palabras que aparecían de forma más significativa con un conjunto de palabras positivas y como negativas aquellas palabras que aparecían con un conjunto de palabras negativas.

2.1.2.6. CLAIRVOYANCE AFFECT LEXICON

Este diccionario fue desarrollado a mano. Las entradas de este diccionario se dividen en cinco campos: el lema de la palabra, part-of-speech, clase afectiva, centralidad e intensidad.

2.1.2.7. ANEW

Se trata de un modelo espacial de varias palabras con carga emocional que contribuye a clasificar textos arbitrarios.

2.2. Servicios Web

Existen múltiples definiciones de servicios web por lo que resulta muy complejo dar una definición de este término. Esta complejidad nos permite hacernos una idea de la cantidad de servicios e implicaciones asociadas que se agrupan bajo este concepto.

En la actualidad, la definición más general de Servicio Web es la que lo define como un conjunto de aplicaciones o tecnologías capaces de interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos con el fin de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. (www.w3c.es/Divulgación/GuiasBreves/ServiciosWeb)

2.2.1. Características de los Servicios Web

Las características deseable de un Servicio Web son:

- Un servicio que debe poder ser accesible a través de la web. Para ello debe utilizar procotolos de transporte estándares como HTTP, y condificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio..

- Un servicio que debe contener una descripción de sí mismo. De esta forma, una aplicación podrá saber cuál es la función de un determinado servicio web.

- Debe poder ser localizo. 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. (http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html)

2.2.2. Tipos de Servicios Web

A nivel técnico, los servicios pueden implementarse de varias formas. En este sentido, podemos distinguir dos tipos de servicios web: los denominados servicios web "grandes", llamados servicios web SOAP, y servicios web RESTful.

- Servicios web SOAP: utilizan mensajes XML para intercomunicarse que siguen el estándar SOAP, un lenguaje XML que define la arquitectura y formato de los mensajes. Dichos sistemas normalmente contienen una descripción legible por la máquina de la descripción de las operaciones ofrecidas por el servicio, escrita en WSDL (lenguaje basado en XML para definir las interfaces sintácticamente).

- Servicios web RESTful: utilizan estándares muy conocidos como HTTP, SML, URI, MIME, y tienen una infraestructura "liger"que permite que los servicios se construyan utilizando herramientas de forma mínima. (http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html)

2.2.3. Arquitectura de los Servicios Web

Los servicios web se componen de varias capas entre las que destacan: servicios de transporte (constituidos por los protocolos de nivel más bajo, que codifican la información independientemente de su formato y que pueden ser comunes a otros servicios), servicios de mensajería, de descripción y de descubrimiento. En las siguiente subsecciones explicaremos más en detalle cada una de estas capas.

2.2.3.1. Servicios de transporte

Capa que se encarga de transportar los mensajes entre aplicaciones. Normalmente se utiliza el protocolo HTTP para este transporte. HTTP es el protocolo de nivel de aplicación más utilizado en la red. Define la sintáxis y

la semántica utilizada para la arquitectura webb. Es utilizado para la transferencia de las transacciones XML a través de la red utilizando los mismos principios de HTML.

2.2.3.2. Servicios de mensajería

Capa encargada de la codificación de los mensajes en XML estándar paraa que pueda así ser interpretado en cualquiera de los nodos de la red. Puede implementar protocolos como XML-RPC o SOAP.

- XML-RPC: Protocolo de llamadas remta que utiliza XML como lenguaje de codificación y HTTP como mecanismo de transporte. Protocolo sencillo, pues solo define algunos tipos de datos y comandos.
- SOAP: Protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes. Es una vía de transmisión entre un SOAP Sender y un SOAP Receiver, pero lo mensajes SOAP deben interactuar con un conjunto de aplicaciones para que se pueda generar un diálogo SOAP.

2.2.3.3. Servicios de descripción

Capa encargada de definir la interfaz pública de un determinad servicio. Esta definición se realiza mediante WSDL, tipo de documento que XML que describe lo que hace un servicio web, dónde se encuentra y la forma de ser invocado.

2.2.3.4. Servicios de descubrimiento

Capa encargada del registro centralizado de servicios, permitiendo que estos sean anunciados y localizados. Para ello se utiliza el protocolo UDDI, el cual define la especificación para construir un directorio distribuido de servicios web, donde los datos se almacenan en XML. Este registro también almacena información sobre las organizaciones que los proporcionan, la categoría en la que se encuentran, y sus instrucciones de uso.

2.2.4. Ventajas e inconvenientes de los servicios web.

Utilizar servicios web puede tener tanto ventajas, como desventajas. A continuación expondremos algunas de ellas.

2.2.4.1. Ventajas

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalan.

- Fomentan los estándares y protocolos basados en texto, de tal manera que facilitad su comprensión y accesibilidad.
- Permiten la combinación e integración de distintos servicios y software independientemente de su ubicación.
- Al apoyarse en HTTP, lo servicios Web se pueden aprovechar de los de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Disminuyen el tiempo de desarrollo de las aplicaciones: Pues gracias a la filosofía de orientación a objetos utilizada, el desarrollo se convierte más bien en una labor de composición.

2.2.4.2. Desventajas

- Bajo rendimiento si se compara con otros modelos de computación distribuida.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear la comunicación entre programas.

2.3. Metodología Scrum

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para su empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

2.3.1. Proceso de desarrollo software

Un proceso, es un conjunto de actividades, acciones y/o tareas que se realizan para crear un producto determinado, en nuestro caso un producto software.

El proceso de desarrollo software, como estructura general, consta de cinco grandes actividades: **comunicación**, **planificación**, **modelado**, **construcción** y **despliegue**. Estas actividades siempre se realizan en este orden; pero dependiendo del modelo que se emplee variará el flujo. Algunos flujos son: lineal, iterativo, evolutivo o paralelo.

El desarrollo de software hace unos años era algo novedoso y tan moderno que no había estrategias claras para crear productos. Por lo que, con motivo de ordenar todo el proceso, se crearon diferentes modelos de proceso. Un modelo de proceso no es más que una estructura para realizar las actividades que forman un proceso.

En primer lugar, se crean los modelos llamados tradicionales, que son procesos muy controlados y con muchas normas y políticas. Se le da mucha importancia a la arquitectura del software. El cliente interactúa con el equipo de desarrollo mediante reuniones separadas en el tiempo. En la primera reunión se fija un contrato que el equipo debe seguir, lo que genera muchos problemas a la hora de desarrollar ya que el equipo tomará decisiones sin tener en cuenta lo que el cliente opinaría al respecto. Podríamos decir entonces que es un modelo impuesto externamente, no solo por el contrato con el cliente sino porque está basado en normas de estándares. Estos modelos se siguen usando en proyectos con equipos grandes y distribuidos. Generan muchos artefactos y documentación.

Más adelante surgieron los modelos de desarrollo ágiles. La agilidad en términos de software se podría definir como la respuesta efectiva al cambio. Pero va más allá, ya que esta idea lleva consigo toda una filosofía determinada en el Manifiesto por el Desarrollo Ágil (Beck et al.,2001). Este Manifiesto propone un nuevo modelo de proceso en el que los individuos e interacciones están por encima de los procesos y las herramientas, el software funcionando sobre la documentación extensiva, la colaboración con el cliente sobre la negociación contractual y la respuesta ante el cambio sobre seguir un plan.

Dado que los modelos ágiles están diseñados para poder realizar cambios en los requisitos en cualquier momento y están pensados para equipos pequeños. Creemos muy conveniente para nuestro trabajo usar una metodología ágil.

2.3.2. Scrum

Scrum es una metodología ágil para gestionar el desarrollo de software. Fue definido por *Ikujiro Nonaka* e *Hirotaka Takeuchi* a principios de los 80. Compararon la forma de trabajo con el avance en formación de scrum (melé en español) de los jugadores de rugby y por esta razón se llama la metodología Scrum. *Nonaka* y *Takeuchi* caracterizan Scrum por el protagonismo de equipos brillantes, auto-organizados y motivados que abordan el desarrollo de sistemas complejos partiendo de una visión general y solapando las fases del desarrollo (Menzinsky, López, Palacio, 2016).

Scrum divide el trabajo en diferentes unidades llamadas *sprints*, que tienen una duración preestablecida de entre dos y cuatro semanas obteniendo siempre al final una versión del software con nuevas prestaciones listas para ser usadas. En cada *sprint* se ajusta la funcionalidad y se añaden nuevas prestaciones priorizando aquellas que aporten más valor.

2.3.2.1. Roles

Esta metodología, hace mucho énfasis en el «equipo de trabajo». Este equipo está formado por diferentes roles:

- **Product Owner**: Representa al cliente. Se encarga de definir los objetivos y de dar valor al producto.
- **Scrum Master**: Encargado de asegurar que se cumplen las buenas prácticas y valores descritos en el modelo Scrum.
- **Scrum Team**: Equipo encargado de desarrollar y entregar el producto. Es autogestionado y multidisciplinar.

2.3.2.2. Artefactos

Los artefactos definidos en Scrum son:

- **Product backlog**: Lista realizada por el Product Owner que contiene los requisitos del producto, ordenados por prioridad. A lo largo del desarrollo crece y evoluciona. En Scrum los requisitos se definen mediante historias de usuarios. Una historia de usuario es la descripción de una funcionalidad que debe incorporar un sistema de software, y cuya implementación aporta valor al cliente.
- **Sprint backlog**: Lista de las tareas que debe realizar el equipo durante el sprint.

2.3.2.3. Eventos

Por último, Scrum también define una serie de eventos:

- **Sprint**: Cada una de las iteraciones del desarrollo.
- **Sprint Planning**: Se trata de la planificación del sprint. Este evento consta de dos partes en las cuales participan el Product Owner y el equipo. En la primera parte el Product Owner presenta lo que quiere que se haga y resuelve dudas; en la segunda parte el equipo estima y deciden hasta donde van a llegar, para posteriormente crear el Sprint Backlog.
- **Daily Sprint Meeting**: Reunión diaria como máximo de quince minutos, de pie, donde cada componente del equipo informa sobre cómo va en sus tareas, lo que hizo el día anterior, lo que hará ese día y los problemas que ha encontrado o los que cree que se va a encontrar.
- **Sprint Review**: Reunión que se realiza al concluir el sprint centrándose en el producto. Se presenta el producto creado en el sprint al Product Owner y este lo analiza.
- **Sprint Retrospective**: Reunión donde se habla de cómo ha funcionado el equipo en el sprint y qué cosas se pueden mejorar para el

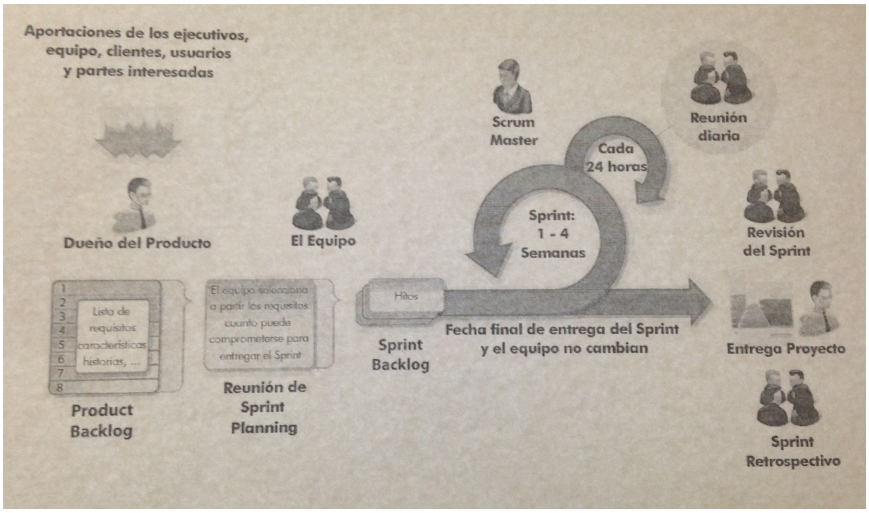

Figura 2.1: Flujo Scrum

siguiente a nivel de proceso o metodología. En esta reunión participa todo el Scrum Team y el Scrum Master.

En la Figura 2.1 podemos ver un diagrama del flujo Scrum. Sería el siguiente: el Product Owner crea el Product Backlog con los requisitos y características por orden de prioridades. A continuación, en un Sprint Planning, se presenta el Product Backlog y se decide qué actividades van a desarrollar y cuánto tiempo van a tardar. Después de esta reunión se elabora el Sprint Backlog con todas las actividades que van a realizar divididas en tareas; cada componente del grupo se asigna una tarea y en cuanto acabe seguirá con la siguiente actividad que no esté hecha. Es muy importante el orden de las tareas, ya que el cliente ha ordenado en el Product Backlog las actividades por prioridad y el equipo desarrollará estas siguiendo dicho parámetro. A continuación, se comienza el sprint con el tiempo establecido. Ni el tiempo ni el equipo que está trabajando se pueden cambiar. Cada día, el equipo se reunirá en una Daily Sprint Meeting para poner en común lo explicado anteriormente. Al final del sprint se realiza la Sprint Review donde se entrega el incremento o el producto hasta el momento al cliente; por lo tanto en esta reunión están presentes el Scrum Team, el Scrum Master y el Product Owner. Por último el Scrum Team junto con el Scrum Master se vuelve a reunir para hacer el Sprint Retrospective.

2.4. Integracion Continua

La práctica de desarrollo de software que se va a utilizar en el desarrollo de este trabajo es la integración continua. Se basa en que los desarrolladores

combinen todos los cambios que realicen en el código en un repositorio común de forma periódica, de tal forma que una vez subidos estos cambios, se ejecutan una serie de pruebas automáticas sobre estos con el fin de validarlos. Aparte de las pruebas automáticas se realizan pruebas manuales para buscar errores más difíciles de encontrar. Hasta que las pruebas sobre el nuevo código no acaban no se puede continuar. Aplicando esto a la metodología Scrum que vamos a utilizar, una historia de usuario no se puede considerar acabada hasta que no pase todas las pruebas, tanto automáticas como manuales.

Las principales ventajas de la integración continua:

- **Detección de errores:** Cada vez que el código cambia se compila y se somete a pruebas para garantizar que no hay errores. Este proceso aumenta la calidad del software y minimiza los riesgos del proceso ya que se tiene control sobre las versiones en todo momento. Cuanto más se tarde en detectar un error más trabajo llevará arreglarlo.
- **Visibilidad del proceso:** Todos los pasos que se realizan en el desarrollo son visibles a todo el equipo, que tiene una estrategia común muy bien definida.
- **Mejora del equipo:** Los desarrolladores no solo tienen una visión muy clara y estructurada del proceso sino que también aprenden a realizar todo tipo de pruebas, lo que les hace mejorar a nivel profesional.

Lo primero para poder utilizar integración continua es tener definido un pipeline, es decir, un conjunto de fases por las que tiene que pasar el software y que están automatizadas. Se establecen criterios para que el código pase de una fase a otra y estrategias para gestionar errores que puedas surgir en las diferentes fases (control de versiones). Es importante tener bien definidas las pruebas que se van a realizar sobre cada fase y que estas puedan garantizar la máxima corrección posible sin tardar más de lo admisible, ya que se necesita un feedback rápido para poder seguir avanzando en el proceso. Cada fase es un grupo de pruebas y cada subida de código es un pipeline distinto que avanza de forma independiente por las fases. Por lo tanto se sabe en todo momento en qué punto se encuentra una versión específica. Esto permite tener una visión general de todo el proceso facilitando notablemente la detección de errores en fases y pipelines concretos.

Para el correcto funcionamiento de esta práctica tiene que haber pequeñas integraciones de forma frecuente, una vez al día por ejemplo. Cuantos menos cambios haya más fácil es la integración en el producto general y solucionar los posibles errores que esta pueda generar. Cabe destacar que aunque una parte de código funcione de forma independiente no implica que vaya a funcionar al integrarlo en un programa más grande, por ello cuanto más frecuentes sean las integraciones mejor.

En nuestro caso la integración continua se aplicará de la siguiente manera:

- **Repositorio:** Se utilizará un repositorio común de *GitHub* en el que se subirán todos los cambios realizados en el código. Al ser un equipo de desarrollo pequeño y estar utilizando la metodología Scrum en principio todos los miembros del equipo estarán trabajando en la misma rama. Esto puede llegar a bloquear el proceso mientras una versión acabe de pasar las pruebas.

- **Jenkins:** Tendremos Jenkins corriendo en un servidor para realizar las pruebas automáticas. Cada vez que se detecten cambios en el repositorio este avisará al servidor que procederá a hacer las pruebas de compilación y funcionamiento sobre el nuevo código.

- **Pruebas manuales:** Algún miembro del equipo realizará las tareas de «tester». Después de las pruebas automáticas se realizará otra serie de pruebas planificadas de antemano para probar mejor la funcionalidad del código. El «tester» no será siempre la misma persona.

- **Pruebas automáticas:** Usaremos un software especial con el fin de controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados. El uso de estas pruebas, nos permite incluir pruebas muy repetitivas y necesarias, dado que habrá pruebas que realizarlas de manera manual nos podrá ser muy costoso.

Capítulo 3

Herramientas

...

...

Resumen: En este capítulo se profundizará en las herramientas que utilizaremos a lo largo del trabajo.

3.1. Diccionario

El diccionario afectivo que vamos a utilizar para marcar las palabras está basado en el diccionario afectivo ANEW traducido al español (Redondo,Fraga,Padrón y Comesaña, 2007). En la traducción de ANEW al castellano participaron 720 estudiantes de Psicología de entre 18 y 25 años, 560 mujeres y 160 hombres. Estos participantes evaluaron las 1034 palabras que contiene ANEW.

Cada palabra debía ser marcada con las dimensiones emocionales: evaluación, activación y control en una escala de 9 puntos.

Cada entrada en el diccionario contiene un número que identifica a la palabra de manera que, esta numeración coincide con el número que dicha palabra tiene en el ANEW; la palabra inglesa (E-word), la palabra original en la base de datos ANEW; la palabra española (S-word) y las evaluaciones afectivas, los valores medios y la desviación estándar para cada dimensión emocional.

Este diccionario, es una adaptación en el que se han elegido las 5 emociones básicas (tristeza, miedo, alegría, enfado y sorpresa) y la neutralidad para no expresar ninguna emoción. Para crearlo participaron 10 personas; cada una de ellas marco cada una de las 1034 palabras con una emoción básica. Cada palabra puede representar varias emociones por lo que se ha

S-Word	Tristeza	Miedo	Alegría	Enfado	Sorpresa	Neutral
abandonado	1,00	0,00	0,00	0,00	0,00	0,00
abejas	0,00	0,50	0,17	0,00	0,00	0,33
aborto	0,67	0,17	0,00	0,00	0,00	0,17

Figura 3.1: Fragmento de la adaptación del diccionario ANEW traducido.

creado un número de confianza c, entre e 0 y el 1, que determina con qué certeza la palabra corresponde a esa emoción; es decir indica el porcentaje de evaluadores que asignaron dicha emoción a la palabra. La suma de estos números debe sumar 1.

Por ejemplo en la Tabla 3.1, podemos ver los valores obtenidos para las palabras «abandonado», «aborto» y «abeja». En este ejemplo se puede ver que tristeza fue la emoción asignada a la palabra «abandonado» por todos los evaluadores, mientras que con «abejas» y «aborto» no hubo acuerdo. En el caso de «aborto», un 67 % de los anotadores le asignaron tristeza, mientras que un 17 % consideró que la palabra no tenía emoción asociada y un 17 % le asignaron miedo.

3.2. Django

Toda la implementación del trabajo se hará en Django (djangoproject.com,2005-2018), un framework para aplicaciones web gratuito y open source escrito en Python.

Django incluye un servidor web que almacena la base de datos que contiene los datos del diccionario y permite hacer las consultas necesarias. La base de datos está formada por instancias del modelo creado, en nuestro caso cada palabra aparecerá modelada mediante su significante y los porcentajes de cada emoción que contiene. Las consultas, por su parte, se realizan mediante una serie de vistas que se crearán según las necesidades funcionales que tengamos y siempre devolverán el resultado en formato JSON.

3.3. Trello

Trello es una aplicación web que permite organizar proyectos y actividades (trello.com,2011-2018). Emplea un sistema kanban para representar las tareas con tarjetas virtuales. Estas tarjetas irán atravesando los distintos artefactos de la metodología scrum con la que estamos trabajando. En la Figura 3.2 podemos ver el estado al inicial del proyecto. Se observa el **Product Backlog**, del que scrum master saca la cantidad de tareas que quiere que se realicen durante el sprint y estas pasan a la lista **To Do**. En la Figura 3.3 se

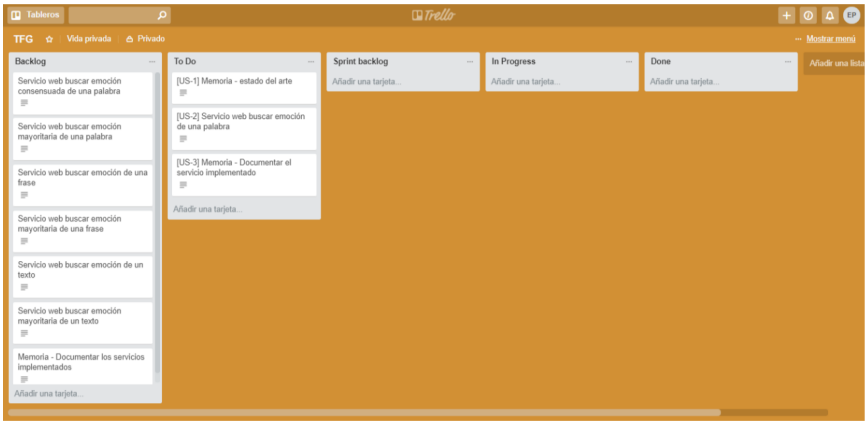

Figura 3.2: Sprint inicial que ilustra la estructura.

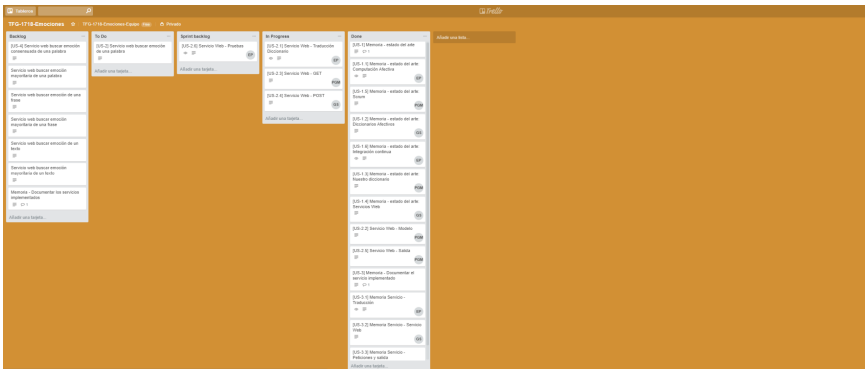

Figura 3.3: Final del sprint inicial.

puede ver un ejemplo más avanzado en el que se puede ver como las tareas has sido divididas en subtareas para formar el **Sprint Backlog**, del que van saliendo en orden para estar **En Progreso** y, una vez acabadas, **Done**.

3.4. Doctest y Jenkins

Como ya se comentó en el capítulo 2.4 utilizaremos Jenkins como parte de la integración continua del proyecto. Esto nos permitirá asegurarnos de que la unificación es correcta y hacer las pruebas automáticas. Esto último se llevará a cabo mediante una orden shell que Jenkins ejecutará para ejecutar el script de pruebas. El script únicamente contendrá las llamadas a los diferentes programas de pruebas que se realicen.

Los programas de pruebas utilizarán Doctest para hacer las pruebas. Doctest es un módulo incluido en la librería estandar de Python. Su fun-

cionamiento se basa en definir la función que se quiera probar y, dentro de un comentario al inicio de esta, poner una serie de llamadas y el resultado que se espera obtener de ellas. Tiene una función «testmod» que realiza las pruebas y devuelve el número de fallos y el resultado de todas las pruebas. Si el número de fallos es mayor que cero el programa de pruebas que haya fallado lanzará una excepción que Jenkins detectará para notificar a todo el equipo que hay algo que va mal.

Notas bibliográficas

Citamos algo para que aparezca en la bibliografía... (Bautista et al., 1998)

Y también ponemos el acrónimo CVS para que no cruja.

Ten en cuenta que si no quieres acrónimos (o no quieres que te falle la compilación en "release" mientras no tengas ninguno) basta con que no definas la constante `\acronimosEnRelease` (en `config.tex`).

En el próximo capítulo...

...

Capítulo 4

Analis del Contenido Afectivo de un Texto

...

...

Resumen: En este capítulo se explicarán los servicios web que se han desarrollado con el fin de analizar el contenido afectivo de un texto. Primero, en la sección 4.1, se presentarán los servicios orientados a identificar la emoción predominante en una palabra concreta, tanto la consensuada como la mayoritaria. Después, en la sección 4.2, se explica como el análisis de la palabra se aplica a determinar la emoción de una frase entera para finalmente, en la sección 4.3, aplicar todo al anális de todo un texto.

4.1. Analis afectivo de una palabra

Como ya se comentó en la sección 2.1.2, para el desarrollo de este trabajo se utilizará un diccionario afectivo en español. Este diccionario, inicialmente en formato Excel, ha sido convertido a un fichero CSV para que resulte más sencillo obtener los datos.

Los datos estarán almacenados en el servidor Django.

- **Modelo:** Los campos que tiene el modelo para cada palabra son la propia palabra, el porcentaje de una determinada emoción que expresan (para las seis emociones que consideramos en nuestro diccionario) y la emoción cuyo porcentaje destaca. Cada una de las palabras que tenemos recogidas se encapsulan en este modelo para ser almacenadas en el servidor.

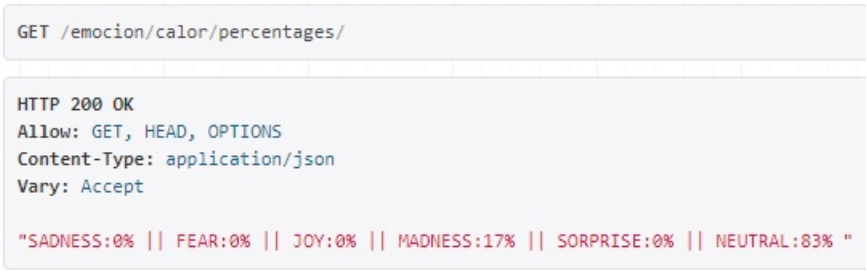

Figura 4.1: Respuesta al buscar los porcentajes de la palabra «calor».

- **Consultas:** Para realizar las difetentes consultas sobre las palabras disponibles se crean una serie de clases que implementan los diferentes métodos de un servicio web REST típico: **GET, POST, DELETE**. Esencialmente **GET**, ya que es el que devuelve información sobre la palabra o alguno de sus campos. Cada una de las diferentes clases o «vistas» aportan una forma diferente de acceder a la información: acceso a toda la lista, a una palabra concreta, a un campo de una palabra concreta, etc...

Una vez que se tiene el servidor y todo el código necesario para el funcionamiento de los servicios web se procede a subir las palabras que recoge nuestro diccionario. Para ello se ha desarrollado un programa en *Python* cuya función es leer el fichero CSV, interpretar cada una de sus líneas y subir la información que obtiene al servidor. Una vez que todas las palabras estén subidas ya se pueden realizar las consultas necesarias.

4.1.1. Servicio web porcentajes

Se trata de un servicio web, implementado en Python que dada una palabra nos devuelve la información respectiva a los porcentajes de cada emoción que posee, mediante una petición **GET** al servidor, y los muestra al usuario.

Como entrada tendríamos la palabra de la cual queremos conocer sus emociones; existirán dos posibilidades, bien que la palabra esté en nuestro diccionario, por lo que la salida sería el porcentaje de cada emoción 4.1, o bien que la palabra no exista en nuestro diccionario, que devolverá un mensaje de «NOT FOUND» 4.2.

4.1.2. Servicio web emoción consensuada

Se trata de un servicio web, implementado en Python que dada una palabra nos devuelve, mediante una petición **GET** al servidor, sí tiene o

```
GET /emocion/te/percentages/

HTTP 404 Not Found
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Not found."
}
```

Figura 4.2: Respuesta al buscar los porcentajes de una palabra que no está en el diccionario.

no emoción consensuada y en caso de tenerla cual. Hablamos de emoción consensuada cuando la confianza de una de sus emociones es 1.

Como entrada tendríamos la palabra de la cual queremos conocer su emoción consensuada, mientras que la salida será la emoción consensuada en caso de tenerla 4.3 o un mensaje de que no la tiene 4.4.

4.1.3. Servicio web emoción mayoritaria

Se trata de un servicio web, implementado en Python que dada una palabra nos devuelve, mediante una petición **GET** al servidor, su emoción mayoritaria. Hablamos de emoción mayoritaria cuando el porcentaje de confianza de la una emoción es mayor al del resto de emociones.

Como entrada tendríamos la palabra de la cual queremos conocer su emoción mayoritaria, mientras que la salida será la emoción mayoritaria 4.5, o mayoritarias (en caso de que haya dos con el mismo porcentaje) 4.6.

```
GET /emocion/bueno/agreed/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

"AGREED: JOY"
```

Figura 4.3: Respuesta al encontrar la emoción consensuada.

4.2. Ánalis afectivo de una frase

...

4.3. Ánalis afectivo de un texto

...

```
GET /emocion/calor/agreed/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

"NO AGREED EMOTION"
```

Figura 4.4: Respuesta si la palabra no tiene emoción consensuada.

```
GET /emocion/calor/main/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

"MAIN: NEUTRAL || %: 83"
```

Figura 4.5: Respuesta al encontrar la emoción mayoritaria.

```
GET /emocion/coqueto/main/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

"MAIN: JOY, NEUTRAL || %: 50"
```

Figura 4.6: Respuesta si hay dos emociones mayoritarias.

Apéndice A

Así se hizo...

...

...

Resumen: ...

A.1. Introducción

...

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

Bautista, T., Oetiker, T., Partl, H., Hyna, I. y Schlegl, E. *Una Descripción de LaTeX 2ε*. Versión electrónica, 1998.

–¿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