Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Utilizando Topic Modeling para buscar términos relacionados

Autor del proyecto: Valerio García Palao

Director del proyecto: María Francisca Pérez Pérez

Yecla, 24 de junio de 2020

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma Fecha

Dedicatoria y Agradecimiento

A mi mujer, Margarita.

Quisiera agradecer en primer lugar a mi directora de proyecto, Paqui Pérez, el brindarme la oportunidad de realizar este proyecto. Ella ha guiado, con infinita paciencia y sabiduría, la realización de este proyecto. Gracias por sus conocimientos aportados y ayuda para conocer este apasionante mundo del aprendizaje automático.

Agradecer a toda mi familia y en especial a mi mujer e hijos, por su constante apoyo y comprensión, por entender que cuando estaba con el proyecto no estaba con todos ellos y es un tiempo que no se puede recuperar.

Para finalizar, agradecer a todos los profesores que durante estos años en la USJ me han impartido clase, conocimiento y "puesta al día" sobre el mundo de la Ingeniería Informática.

Tabla de contenido

Resumen

Es bastante frecuente encontrar estudios sobre la aplicación de modelos de aprendizaje automático en el ciclo de vida del desarrollo software, basadas en los textos de su código fuente y cuyo fin es obtener información sobre el producto. Estos textos están alojados en repositorios de software u otros sistemas de almacenamiento, siendo una de las mayores fuentes de información y estudio para la Ingeniería de Software, pero con un tamaño de datos en crecimiento constante. Resulta necesario el uso de estos modelos de aprendizaje para su procesamiento, obtención de información y generación de nuevos conocimientos.

El objetivo del proyecto es implementar una herramienta que utilice la técnica Topic Modeling para el análisis de texto de código de software y realizar búsquedas de términos relacionados se proporcionan como consulta. El desarrollo del proyecto incluye, qué es la técnica aplicada, un algoritmo basado en ella, una librería que implementa el algoritmo y el desarrollo de la herramienta. El resultado del proyecto es una herramienta creada como plug-in de Eclipse, que aplica Topic Modeling, utiliza como entrada software especificado (código fuente o modelos software) y una consulta con términos. Como salida genera topics y la relación con la consulta especificada. Para la gestión del proyecto se emplea la metodología ágil Scrum porque se valora la respuesta rápida a cambios, participación del cliente y prototipos en constante evolución.

Palabras clave: Topic Modeling, topics, modelos software, aprendizaje automático, Scrum.

Abstract

It is quite common to find studies on the application of machine learning models in the development software life cycle, based on the texts of its source code and whose purpose is to obtain information about the product. These texts are housed in software repositories or other storage systems, being one of the largest sources of information and study for Software Engineering, but with a constantly growing data size. The use of these learning models is necessary for their processing, obtaining information and generating new knowledge.

The goal of the project is to implement a tool that uses the Topic Modeling technique for text analysis of software code and searching for related terms are provided as a query. The development of the project includes, what is the applied technique, an algorithm based on it, a library that implements the algorithm and the development of the tool. The result of the project is a tool created as an Eclipse plug-in, which Topic Modeling applies, using specified software (source code or software models) and a query with terms as input. As output, it generates topics and the relation with the specified query. For the project management, the agile Scrum methodology is used because the rapid response to changes, customer participation and constantly evolving prototypes are valued.

Key words: Topic Modeling, topics, software models, machine learning, Scrum.

1. INTRODUCCIÓN.

Obtener valor de la información para generar conocimiento es un fin que se está persiguiendo continuamente por parte de las personas y es debido a la necesidad de generar cambios para evolucionar. Se hace imprescindible el desarrollo de técnicas, herramientas, métodos, ciencia e ingeniería que permiten el procesamiento de los datos para generar una información consistente y orientada al aumento de conocimiento. A la misma vez, la expansión del conocimiento adquirido hace que las fuentes de datos sean cada vez más grandes, precisas, completas y generen nuevas fuentes también, haciendo que la espiral de conocimiento vaya en aumento.

Desde mediados del siglo pasado se empieza a acuñar una ciencia llamada "Ciencia de Datos – Data Science" cuyo fin es realizar estudios sobre la información para obtener datos que sirvan para el aumento del conocimiento. Los datos por si solos ya no tienen validez, como explica Upadhyaya (1), además el estudio de ellos cada vez es más complejo conforme se ha simplificado su generación, acceso y demanda sobre ellos de más información. La Ciencia de Datos viene a explotar esos datos, con sus diferentes ramas o áreas, permitiendo realizar estudios desde un punto de vista más complejo, generando resultados a cuestiones que antes eran difíciles de obtener o impensables. Aparecen ramas derivadas de esta ciencia como son: Aprendizaje Automático, Minería de Datos, Procesamiento del Lenguaje Natural, etc. y permiten la aparición del concepto que Loukides (2) llama "producto de datos – data products", es decir, mediante la ciencia de datos se generan productos derivados de los datos que dan mayor accesibilidad al conocimiento e interpretación humana. Así, el dato se convierte en la unidad mínima sobre la que generar conocimiento que tendrá impacto en el negocio, al tomar decisiones con confianza capaces de generar valor comercial.

Un campo de actuación de la Ciencia de Datos es la Minería de Textos, con el fin de extraer y procesar masas de documentos, descubriendo conocimiento que ningún documento tiene por sí sólo, pero no deducirlo al carecer de sistemas de razonamiento. Asociada a esta área se encuentra el modelo o técnica de Topic Modeling. Ella se encarga de obtener "temas" o "topics" presentes en los textos de diferente tipo: libros, reseñas, artículos, elementos de redes sociales y también de textos de código software. Blei (3) y Navarro (4) precisan que son una serie de algoritmos capaces de detectar y extraer relaciones latentes (ocultas) de grupos de documentos o textos expresándose en topics. Éstos son conjuntos de palabras que tienden a aparecer juntas en los mismos contextos.

Una de las fuentes de información para la Ingeniería de Software son los repositorios de código fuente de software. Existen bastantes estudios realizados en los últimos años que contemplan la aplicación de modelos de aprendizaje automático en todo el ciclo de vida del desarrollo software con la finalidad de obtener caracterizaciones sobre el producto, basadas en

los textos de su código fuente. Topic Modeling puede mejorar la comprensión y análisis de grandes cantidades de textos de código fuente en repositorios de software con diferentes niveles de granularidad. Es utilizada hoy en día en la Ingeniería de Software para obtener conocimiento y su aplicación en áreas como requisitos, desarrollo, calidad y pruebas. Pero también para la gestión de proyectos y la interacción entre equipos, generando consultas que permiten descubrir temas latentes en el código y extraer relaciones con los documentos, localización de modelos, similitud entre autores, complejidad y evolución del software.

El objetivo principal del PFG es demostrar el uso de Topic Modeling para analizar textos de código fuente y modelos de software buscando términos relacionados con el texto, para posteriormente mostrar el valor de pertenencia de la cadena de búsqueda a los documentos del corpus. De esta forma, los términos relacionados permiten localizar de forma automática, en un gran espacio de búsqueda, fragmentos de código o fragmentos de modelos que están relacionados con los términos buscados. Para ello, se implementa una herramienta (dónde se aplicará Topic Modeling sobre textos de código fuente y modelos de software) que permite realizar consultas sobre el resultado obtenido por la técnica. Los resultados pueden ser especialmente útiles en tareas de mantenimiento software, que requieren localizar el fragmento de software a actualizar. Entre las tareas de mantenimiento software más relevantes se encuentran la localización de características o bugs. Esta tarea en entornos industriales es una tarea compleja por la cantidad de software que se acumula durante años.

Para la ejecución del proyecto se emplea la metodología Scrum, encuadrada dentro de las metodologías ágiles y permite un desarrollo incremental, solapamiento de fases de desarrollo y participación del cliente en todos los procesos.

1.1. Organización del proyecto.

Esta memoria está estructurada de la siguiente forma: el capítulo 2 es una presentación del estado del arte de Topic Modeling. Tras una investigación y estudio en profundidad, se muestra qué es lo que hay hecho en este campo. En el capítulo 3 se exponen los objetivos del proyecto, tanto principales como secundarios. La metodología empleada para conseguirlos y un plan de trabajo están contenidos en el capítulo 4. El diseño y desarrollo de la aplicación que utiliza Topic Modeling, junto con la herramienta empleada, se contempla en el capítulo 5. El capítulo 6 presenta el presupuesto económico necesario para la realización del proyecto. En el capítulo 7 se expondrán los resultados obtenidos al finalizar el proyecto, así como las desviaciones aparecidas y su corrección. Finalmente, en el capítulo 8, concluye el documento resumiendo el grado de cumplimiento de los objetivos y trabajos futuros.

2. ANTECEDENTES. ESTUDIO DEL ARTE.

En este capítulo se explican, la evolución tecnología y científica relacionada con Topic Modeling; cómo se llega hasta esta técnica y su aplicación al análisis de software. Para ello se comentan las acciones que hay realizadas en el campo de la Ciencia de Datos hasta su relación con Topic Modeling, generando una breve hoja de ruta para ello.

2.1. De la Ciencia de Datos a Topic Modeling.

La pregunta generalista es ¿por qué hay una necesidad de una ciencia de datos? Autores como Matuszek (5) explican que la gran cantidad de documentación actual ha originado nuevos enfoques técnicos para poder realizar su explotación y obtener la información que contiene y por tanto generar conocimiento. Al fin y al cabo, es lo que se persigue: conocimiento para generar valor. El estudio de Gardner (6) es bastante claro sobre los motivos de la necesidad, relacionándolo a cuatro tendencias: crecimiento de la masa de datos, mayor conexión entre dispositivos y fuentes de orígenes de datos, una demanda de datos, casi rozando la dependencia de ellos, y la económica. Aunque Upadhyaya (1) también apoya parte de las tesis de Gardner explicando que los datos aislados ya no tienen sentido y que a la misma vez se vuelve más complejo su estudio conforme su acceso es más fácil. La "*ciencia de datos o data science"* como comenta Neri (7): "*es la teoría y método del estudio eficiente de los datos y sus fenómenos*" cuyo objetivo final es obtener conocimiento de los datos para ayudar a tomar decisiones o acciones. Este mismo autor también aporta otra definición más concreta: "*es el estudio de la eficiencia en el descubrimiento y/o optimización de patrones, procesos, modelos y propiedades de flexibles conjuntos de datos que conllevan una alta complejidad computacional*" con el mismo objetivo de la anterior. Es relevante la mención a patrones y modelos sobre el conjunto de los datos.

Otras definiciones más ajustadas o detallistas son las dadas por: Rivera (8): "*es la extracción de conocimiento de grandes volúmenes de datos estructurados y no estructurados mediante el uso de técnicas de minería de datos y análisis predictivo*", detallando datos importantes como que ya no importa cómo estén los datos y el uso ya de técnicas específicas. Bhatia y Kaluza (9) "*la ciencia de datos abarca todo el proceso de obtención de conocimiento mediante integración de métodos de estadística, informática y otros campos para obtener información de datos. En la práctica, la ciencia de datos abarca un proceso iterativo de recolección de datos, limpieza, análisis, visualización y despliegue*" dónde se especifica que hay ciencias e ingenierías asociadas y qué procesos hay que realizar para la obtención de resultados. Pero Loukides (2) dónde señala que "data science enable the creation of Data Products", es decir, la ciencia de datos genera "productos de datos", mucho más accesibles al conocimiento y la interpretación humana. ¿Cómo se generan esos productos datos y cuál es la

relación con el proyecto? La ciencia de datos provee áreas o técnicas específicas para ello. En el caso de este proyecto ya se ha definido la utilización de la técnica de Topic Modeling.

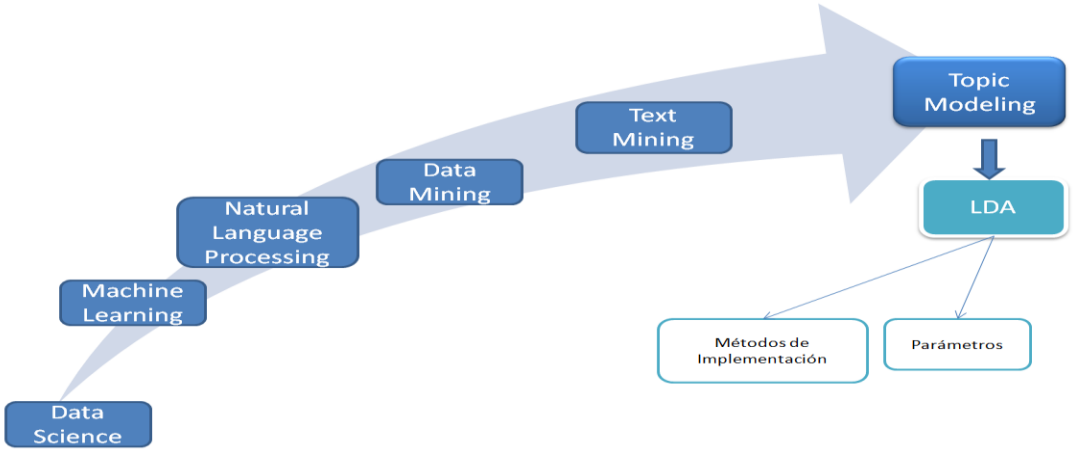


Figura 1: Hoja de ruta desde Ciencia de Datos a Topic Modeling.

Figura 1 detalla la hoja de ruta partiendo desde los orígenes de la Ciencia de Datos hasta llegar a Topic Modeling. Es verdad que la Ciencia de Datos tiene más áreas, pero este diseño de hoja de ruta sirve para poner en antecedentes el origen de Topic Modeling y su uso en el proyecto. Para extraer ese conocimiento se aplica Topic Modeling. Éste, según Navarro (4) o Díaz (10), es la técnica que permite la identificación de patrones recurrentes en colecciones de texto y por tanto es una forma concreta de hacer Text Mining. La minería de textos es calificada, según Eito y Senso (11), como la hermana pequeña de la minería de datos y se asocia a ella como una evolución al estar asociada, desde sus inicios, a la lingüística computacional pero con la característica que trabaja, al contrario que la minería de datos, sobre repositorios de textos, con información no estructurada. Estas dos minerías, a su vez han derivado del procesamiento de lenguaje natural, puesto que éste evolucionó, no sólo siendo capaz de actuar sobre lenguas en código formal, sino que se amplió su marco de influencia al incluir registros que no siguen la norma de la lengua escrita. A su vez el procesamiento del lenguaje natural hace uso de algoritmos de aprendizaje automático y dentro de este campo, se pueden encuadrar los diferentes tipos métodos de aprendizaje: supervisado, no supervisado y de refuerzo. En el aprendizaje no supervisado, no se cuenta con esos datos de entrenamiento que ayudan a discernir entre la información adecuada o no. Su objetivo principal es encontrar patrones ayudado por técnicas estadísticas. El método deberá ser capaz de encontrar relaciones entre los datos de las diferentes instancias (documentos) y organizarlos (agruparlos) de alguna forma. Carreño (12) describe que dicha técnica es utilizada en diferentes áreas como la minería de datos, aprendizaje automático, reconocimiento de patrones o biomedicina, es decir al utilizarse en minería de datos, Topic Modeling tiene una base de aprendizaje no supervisado.

Tras el análisis de la hoja de ruta puede definirse Topic Modeling como un conjunto de algoritmos, encuadrados dentro del campo de la minería de textos y con el objetivo de descubrir "topics" o "temas" ocultos en los documentos de un corpus. Blei (13) describe el funcionamiento como sigue: "*Topic Modeling toma una colección de textos como entrada. Descubre un conjunto de topics – temas recurrentes que se discuten o comentan en la colección de documentos- y el grado en que cada documento exhibe esos temas*". Además, comenta que el modelo brinda un marco en el que explorar y analizar los textos, pero sin tener que aplicar ni decidir los temas por adelantado ni codificar minuciosamente cada documento de acuerdo con ellos. Para la realización Blei propone el modelo "Latent Dirichlet Allocation" (LDA), que es un modelo probabilístico generativo de un corpus, basado en la distribución de Dirichlet. Blei y Lafferty (14) describen LDA como el modelo más simple de Topic Modeling y sobre el cual están desarrollándose otra serie de modelos desde su aparición. LDA se utiliza como modelo de procesamiento del corpus en el proyecto.

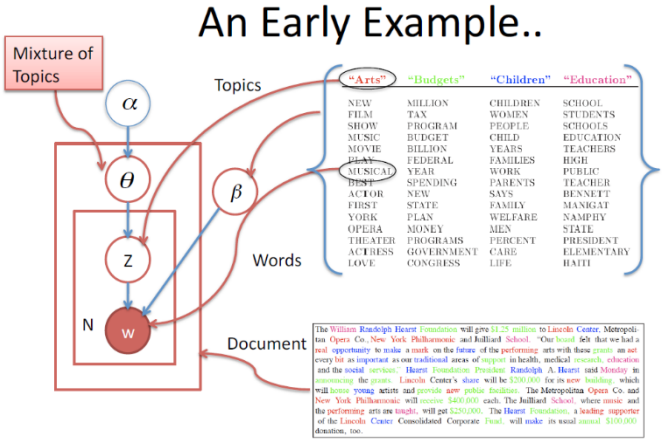


Figura 2. Relación entre representación gráfica LDA con un ejemplo de texto y topics. Santhanam (15).

2.2. Topic Modeling y Análisis de Software.

La funcionalidad del objetivo del proyecto es que se utilice Topic Modeling para analizar términos utilizados en software, en código fuente o modelos de software. El análisis se realizará mediante la búsqueda de términos relacionados con el texto, para posteriormente mostrar el valor de probabilidad de pertenencia de la cadena de búsqueda a ese documento. Esta sección muestra evidencias de uso de Topic Modeling para el análisis de textos de código software. Para ello se detalla literatura que relaciona las necesidades en la ingeniería de software para el uso de Topic Modeling.

Blei (13) formula preguntas asociadas a la explotación y análisis de los datos, pero también buscando ir un paso por delante, comenta que el mundo del Topic Modeling entra dentro de un campo mayor llamado "modelado probabilístico", proporcionando un lenguaje para expresar suposiciones sobre datos y métodos genéricos para computar con esas suposiciones. Afirma que LDA es un modelo probabilístico con variables ocultas, pero que ese proceso generativo para componer la estructura oculta del topic, como las palabras observadas en los textos, no deja de ser lo que se llama "inferencia probabilística". En resumen, este

modelo ayuda a interpretar y comprender textos, pero es trabajo de las personas utilizar, interpretar y comprender la información ofrecida. Se recuerda que es un método no supervisado. El propio modelado probabilístico da un lenguaje para articular suposiciones sobre sus datos y algoritmos para calcular las suposiciones sobre archivos grandes, pero no para hacer valoraciones por su propia cuenta. Silvestre (16) comenta una pequeña relación con el análisis de software, aunque es muy generalista comentando que se ha de tener en cuenta que, dependiendo del campo de aplicación, se debe emplear un conjunto de documentos más específico o genérico. Por ejemplo, para organizar estudios del campo de la ingeniería del software, previamente se deberá hacer una selección de documentos que pertenezcan a este campo para, después, aplicando algún algoritmo de Topic Modeling, hacer una separación más selectiva.

Se tiene entonces una base justificativa para poder investigar cómo aplicar el método al estudio del software y ver cómo ese análisis sirve para resolver problemas de ingeniería de software: análisis de requisitos, predicción de fallos, costes y estimaciones de esfuerzo, mantenimiento y reingeniería, etc.

Agrawal et al. (17) van un paso más y en su estudio presentan carencias de Topic Modeling, pero aplicado al proceso de análisis del software con datos no estructurados. En la parte positiva abogan que esta forma de análisis supone un gran cambio en la forma de interpretar y analizar el software. Es de interés la tabla que generan (tabla 2 de su estudio) y que se refleja en la Figura 3. Presuponen que no existe ningún modelo sobre el orden de los datos y su objetivo es demostrar que LDA se ve influido por los efectos de orden de los textos, documentos, etc. Pero también demuestran la utilidad (segunda columna a la derecha de la tabla "*Task/Uses Cases*") de Topic Modeling al emplearlo para esos casos de uso descritos. Además, concluyen, entre muchos efectos de ajuste para LDA, que el campo de la analítica de software necesita hacer mucho más uso de la "ingeniería de software basada en búsquedas (SBSE)" puesto que se ha demostrado la utilidad en casi todas las áreas de la ingeniería del software. Dan ejemplos de cómo las búsquedas y sus ajustes ayudan significativamente a la predicción de defectos. En cambio, Binkley et al. (18) crean un artículo para entender LDA en base al análisis del software. Su principal argumento es que han detectado un uso creciente en la comprensión del código fuente y los artefactos relacionados con el software, principalmente debido al poder de modelado que posee el método. Avisa que tiene un coste sensible, que es el ajuste de los parámetros del modelo, pero añade que la configuración más adecuada depende de tres piezas fundamentales: cómo resolver el problema o qué problema a resolver, el corpus de entrada y las necesidades de la ingeniería a la hora de realizar el análisis. Aboga por el método LDA también porque es capaz de identificar patrones tanto dentro del código como entre el código y sus artefactos relacionados. Agrawal et al. (17) demuestran, años más tarde, que con una mejora sobre LDA y ajuste de los parámetros las mejoras son importantes.

REF	Year	Citations	Venues	Mentions instability in LDA?	Uses Default Parameters	Does tuning?	Conclusion	Tasks / Use cases	Unsupervised or Supervised
[5]	2011	112	WCRE	Y	Y	N	Explored Configurations without any explanation.	Bug Localisation	Unsupervised
[6]	2010	108	MSR	Y	Y	N	Explored Configurations without any explanation. Reported their results using multiple experiments.	Traceability Link recovery	Unsupervised
[7]	2014	96	ESE	Y	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Stackoverflow Q&A data analysis	Unsupervised
[4]	2013	75	ICSE	Y	Y	Y	Uses GA to tune parameters. They determine the near-optimal configuration for LDA in the context of only some important SE tasks.	Finding near-optimal configurations	Semi-Supervised
[8]	2013	61	ICSE	Y	Y	N	Explored Configurations without any explanation.	Software Requirements Analysis	Unsupervised
[9]	2011	52	MSR	Y	Y	N	They validated the topic labelling techniques using multiple experiments.	Software Artifacts Analysis	Unsupervised
[10]	2014	44	RE	Y	Y	N	Explored Configurations without any explanation.	Requirements Engineering	Unsupervised
[11]	2011	44	ICSE	Y	Y	N	Open issue to choose optimal parameters.	A review on LDA Mining software repositories using topic models	Unsupervised
[12]	2014	35	SCP	Y	Y	N	Explored Configurations without any explanation.	Software Artifacts Analysis	Unsupervised
[13]	2012	35	MSR	Y	Y	N	Choosing the optimal number of topics is difficult.	Software Defects Prediction	Unsupervised
[14]	2014	31	ESE	Y	Y	N	Choosing right set of parameters is a difficult task.	Software Testing	Unsupervised
[15]	2009	29	MSR	Y	Y	N	Explored Configurations without any explanation and accepted to the fact their results were better because of the corpus they used.	Software History Comprehension	Unsupervised
[16]	2013	27	ESEC/FSE	Y	Y	Y	Explored Configurations using LDA-GA.	Traceability Link recovery	Supervised
[17]	2014	20	ICPC	Y	Y	N	Use heuristics to find right set of parameters.	Source Code Comprehension	Unsupervised
[18]	2013	20	MSR	Y	Y	N	In Future, they planned to use LDA-GA.	Stackoverflow Q&A data analysis	Unsupervised
[19]	2014	15	WebSci	Y	Y	N	Explored Configurations without any explanation.	Social Software Engineering	Unsupervised
[20]	2013	13	SCP	Y	Y	N	Their work focused on optimizing LDAs topic count parameter.	Source Code Comprehension	Unsupervised
[21]	2012	13	ICSM	Y	Y	N	Explored Configurations without any explanation.	Software Requirements Analysis	Unsupervised
[22]	2015	6	IST	Y	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software re-factoring	Supervised
[23]	2016	5	CS Review	Y	Y	N	Explored Configurations without any explanation.	Bibliometrics and citations analysis	Unsupervised
[24]	2014	5	ISSRE	N	Y	N	Explored Configurations without any explanation.	Bug Localisation	Semi-Supervised
[25]	2015	3	JIS	Y	Y	N	They improvised LDA into ISLDA which gave stability across different runs.	Social Software Engineering	Unsupervised
[26]	2015	2	IST	Y	Y	Y	Explored Configurations using LDA-GA.	Software Artifacts Analysis	Supervised
[27]	2016	0	JSS	N	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software Defects Prediction	Unsupervised

Figura 3. Resultados aplicación uso de TM en textos software. Agrawal et al. (17).

Linstead et al. (19) sí centran su opinión favorable para el problema de la minería de grandes repositorios de software en múltiples niveles de granularidad, resaltando que sus enfoques les permiten descubrir automáticamente temas o topics incrustados en el código y extraer las relaciones de documentos por topic y distribuciones por topic y autor, además de servir como un resumen sobre el contenido del software y las actividades de desarrollo, proporcionándoles información estadística para el análisis del software, mantenimiento, similitudes en desarrollos y evolución de los topics en la vida del software durante el tiempo.

López-Nozal et al. (20) avanzan más en el uso y lo relacionan con la gestión de proyectos, esgrimiendo que la caracterización tanto de productos como de interacciones basadas en texto es el primer paso para comprender y hacer más eficiente la gestión del proyecto. Argumentan que Topic Modeling mejora la compresión de grandes cantidades de textos asociados al software mejorando los procesos de calidad de todo el ciclo de vida. Inciden en el gran interés despertado por la aplicación de estos métodos y técnicas en la comunidad científica e industrial.

Muchos autores han utilizado herramientas que aplican Topic Modeling y/o LDA que son verdaderos asistentes para los ingenieros de software. Algunas herramientas son: IMTviz de

Saeidi et al. (21) que presentan un esquema de arquitectura que puede servir como base para el desarrollo de la herramienta de este proyecto.

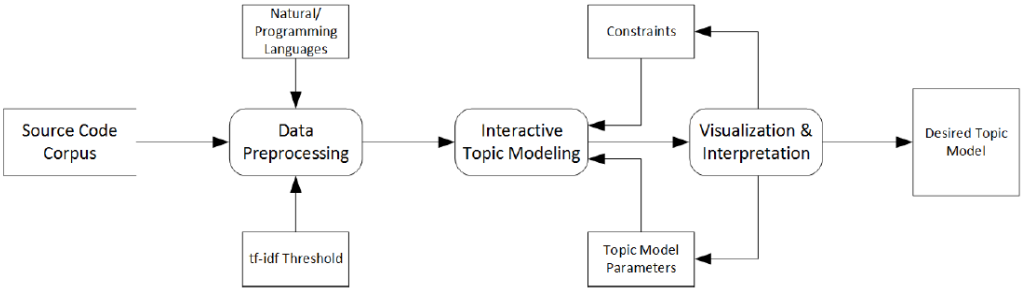


Figura 4. Arquitectura básica de ITMViz. Saeidi et al. (21).

O "Java Software Engineers Assistant" (JSEA) de Wang y Liu (22) que la utilizan para los equipos de mantenimiento de software. Asunción et al. (23) presentan tres herramientas: "TRASE Tool", "ACTS Tools" y "TEAM Tool" como un conjunto completo de análisis y visualización. Bissyandé et al. (24) presentan a "ORION" un proyecto de un motor de búsquedas de software integrado con diversos artefactos de software y "TopicNets" de Gretarson et al. (25) que muestran una herramienta web para visualizar e interactuar con los topics del corpus analizado. Pero son herramientas generalistas, cerradas, con consultas limitadas y no permiten hacer exploraciones sobre fragmentos de texto. Sin embargo, la aplicación desarrollada tiene lo anterior y es exportable, centrada en las consultas y con entrada de datos en diferentes formatos.

En resumen, se llega a la conclusión, tras las evidencias anteriores, que Topic Modeling puede emplearse para el análisis de textos de código y modelos de software y por tanto de aplicación al objetivo del proyecto. También se ha comprobado que Topic Modeling está adquiriendo cada vez mayor importancia en la ingeniería del software, con usos muy diversos, que abarcan desde la búsqueda de requisitos a nivel de consultas a revisiones de bugs en código fuente, pasando por los mantenimientos de carácter predictivo o correctivo, con lo cual es un candidato idóneo para la aplicación en el proyecto.

Para finalizar el estudio del estado del arte sobre el análisis de información en la ingeniería del software, diferentes autores, Lapeña et al. (26), Pérez et al. (27) o Viet Phan et al. (28), utilizan otras técnicas de recuperación diferentes, empleando otros algoritmos como LSI, LSA, hLDA, SLDA o metodologías diferentes, sin embargo, no exploran Topic Modeling como técnica de recuperación de fragmentos de código fuente o fragmentos de modelos de software.

3. OBJETIVOS

Este capítulo presenta los diferentes objetivos, en principales y secundarios, que va a tener el proyecto.

3.1. Principales o generales.

Este PFG se centra en utilizar Topic Modeling para buscar términos relacionados.

Se establecen los siguientes objetivos principales, según propuesta de proyecto presentada en la siguiente tabla:

Tabla 1. Objetivos principales del proyecto.

CÓDIGO	OBJETIVOS PRINCIPALES
OP-01	Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado.
OP-02	Implementar una aplicación en Java que utilice la técnica Topic Modeling.

3.2. Secundarios o específicos.

Los objetivos secundarios se presentan en la siguiente tabla:

Tabla 2. Objetivos secundarios del proyecto.

CÓDIGO	OBJETIVOS SECUNDARIOS
OS-01	Investigar y estudiar el estado del arte desde Data Science a Topic Modeling para comprender la técnica a utilizar.
OS-02	Establecer los fundamentos teóricos y conceptuales de Topic Modeling y LDA para su uso en la aplicación a desarrollar.
OS-03	Establecer la relación de Topic Modeling con el Análisis de Textos de Software.
OS-04	Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.
OS-05	Analizar la estructura de datos que se reciben en la técnica para posteriormente aplicar Topic Modeling, muestreo y ver relaciones entre términos.
OS-06	Establecer escenarios en los que se implementará y analizará la aplicación Java para evaluar su comportamiento
OS-07	Evaluar comportamiento de Topic Modeling en el conjunto de datos del corpus seleccionado para ver resultados y establecer conclusiones

OS-08	Investigar y aprender sobre metodología SCRUM para aplicar al desarrollo del proyecto
OS-09	Montar infraestructura hardware y software para desarrollo de la aplicación y su evaluación.

4. METODOLOGÍA.

Según Javier Garzás en la entrevista con Alores (29), del blog de Velneo, existe una percepción generalizada, entre las empresas de desarrollo de software, de la necesidad de cambiar de un modelo tradicional en cascada industrial a otras metodologías para no quedarse fuera del mercado; modelos centrados en las personas, en la creatividad, la agilidad y los cambios constantes. Esto es lo que promueven las llamadas metodologías ágiles, como programación extrema (XP), Kanban, Lean, Agile o Scrum.

El comentario anterior de Garzás es debido a un cambio constante en la demanda del mercado, que junto a la triple necesidad de reducción de: presupuesto, plazo de servicio y errores; está generando una aplicación de nuevas metodologías de desarrollo de software, así como de la gestión de proyectos. El beneficio final para la empresa va a ser una mayor competitividad e integración del cliente en su modelo de negocio, una disminución de costes y tiempo de producción controlado y ajustado, junto con una mejora de la calidad de producto y aceptación por parte del cliente. Para el usuario final representará productos o servicios adaptados a sus necesidades, en diseño, valor, calidad, sensaciones y experiencias.

Para desarrollar este PFG, se utiliza la metodología ágil, concretamente SCRUM. Obteniendo así un enfoque más adaptado a cambios, participación del cliente, prototipos tempranos, etc. Como en este proyecto, el equipo de desarrollo está formado por un solo miembro (proyectando), este capítulo describe cómo se ha adaptado SCRUM, sin perder sus elementos esenciales, al proyecto; dónde se requiere una retroalimentación por parte de la Dirección de Proyecto constante; dónde es necesario obtener resultados de una forma continua y poder realizar alteraciones o correcciones según sea adecuado para el producto o proyecto.

Se hubiera podido utilizar otras metodologías tradicionales, dentro de la gestión de proyectos predictiva existen bastantes, incluso alguna de carácter personal como (PSP[SM]) "Proceso Software Personal"[1] (30) (31) (32), pero al partir de una duración limitada del proyecto, establecida de antemano, se opta por esta metodología para presentar la ejecución del proyecto de una manera más ágil, minimizando riesgos, al comprobar con bastante frecuencia el avance del mismo; no tener que planificar detalladamente y anticipadamente como en la gestión predictiva, avanzando así más rápidamente, al dividir las tareas en pequeños problemas, teniendo mayor capacidad de reacción y captación de las funcionalidades requeridas.

1 PSP propone a los ingenieros software una forma disciplinada de estructurar su trabajo personal, dando un conjunto de prácticas de uso personal para la gestión del tiempo y mejora de la productividad para las tareas de desempeño de desarrollo y mantenimiento de sistemas.

4.1. ¿Qué es la metodología ágil?

A mediados de los años 90 del siglo pasado, cuando empiezan a establecerse metodologías de calidad total, de lean manufacturing, de Kanban, etc. Los ingenieros de software se cuestionan si el desarrollo de software, como en otros trabajos basados en el conocimiento, se puede producir con patrones de procesos industriales. También se empieza a aceptar que el conocimiento tácito de la persona que realiza el trabajo puede aportar más al valor al resultado que la tecnología y los procesos empleados, junto con la aplicación de técnicas de trabajo en grupo, aumentando así el conocimiento de las acciones a realizar para dar más valor al trabajo realizado.

En marzo de 2001, 17 profesionales del software fueron convocados por Kent Beck[2], teniendo todos ellos algo en común: eran críticos de los modelos de producción basados en procesos Palacio (33). En la reunión se acuñó el término "métodos ágiles" para definir a aquellas metodologías que estaban surgiendo como alternativa a las metodologías formales, que consideraban excesivamente pesadas y rígidas por su carácter normativo y con fuerte dependencia de planificaciones detalladas previas al desarrollo. Estos integrantes de la reunión, resumieron en cuatro postulados lo que ha quedado denominado como "Manifiesto Ágil", que son los valores sobre los que se cimentan estos métodos. Dichos principios son los siguientes Palacio (33):

«Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a ***valorar****:*

- ***A los individuos y su interacción****, por encima de los procesos y las herramientas.*
- ***El software que funciona****, por encima de la documentación exhaustiva.*
- ***La colaboración con el cliente****, por encima de la negociación contractual.*
- ***La respuesta al cambio****, por encima del seguimiento de un plan.*

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.»

Dichos principios tienen la siguiente explicación básica según los autores Palacio (33) y Menzinsky et al. (34):

- *A los individuos y su interacción*. En la agilidad los procesos sólo son un soporte para guiar el trabajo no el fin de consecución de los objetivos.
- *El software que funciona*. El producto es lo que tiene valor por sí mismo para el cliente y el proveedor. Anticipar cómo funcionará el producto final observando prototipos y partes ya terminadas supone una realimentación estimulante y enriquecedora, que genera ideas difíciles de concebir en un primer momento.

2 Ken Beck había publicado un par de años antes el libro en el que explicaba la nueva metodología de desarrollo llamada "Extreme Programming" (XP).

- *La colaboración con el cliente*. El objetivo de un proyecto ágil no es controlar la ejecución para garantizar que los planes iniciales se cumplen, sino proporcionar de forma continua el mayor valor posible al producto. El cliente debe acompañar en todo el proceso al equipo de desarrollo.
- *La respuesta al cambio*. A diferencia de la gestión predictiva, dónde se necesita planificar y controlar para el cumplimiento del plan de trabajo, en la gestión ágil, los principios son la anticipación y adaptación, teniendo así una capacidad de respuesta alta ante los cambios que pueden surgir debido a requisitos o funcionalidades que no son estables.

Tras acabar la conferencia, el grupo de trabajo aumentó los postulados anteriores y redactó lo que se conoce como un "Manifiesto Ágil"[3] con 12 principios, sobre los cuales se desarrollan las diferentes metodologías de trabajo ágiles. Estas metodologías o modelos van conformando lo que se conoce como "gestión evolutiva" en contraposición con la "gestión predictiva". Autores como Palacio (35) y Palacio (33) ilustran las diferencias entre las dos formas de gestionar como puede verse en la Figura 5.

En dicha figura se muestran las diferentes características entre la gestión de proyectos ágil o evolutiva y la tradicional o predictiva, distinguiéndose en la gestión ágil entre la ingeniería concurrente y la agilidad. En esta última, la diferencia estriba en el cambio de conocimiento por los procesos de la concurrente al conocimiento basado en las personas de la agilidad.

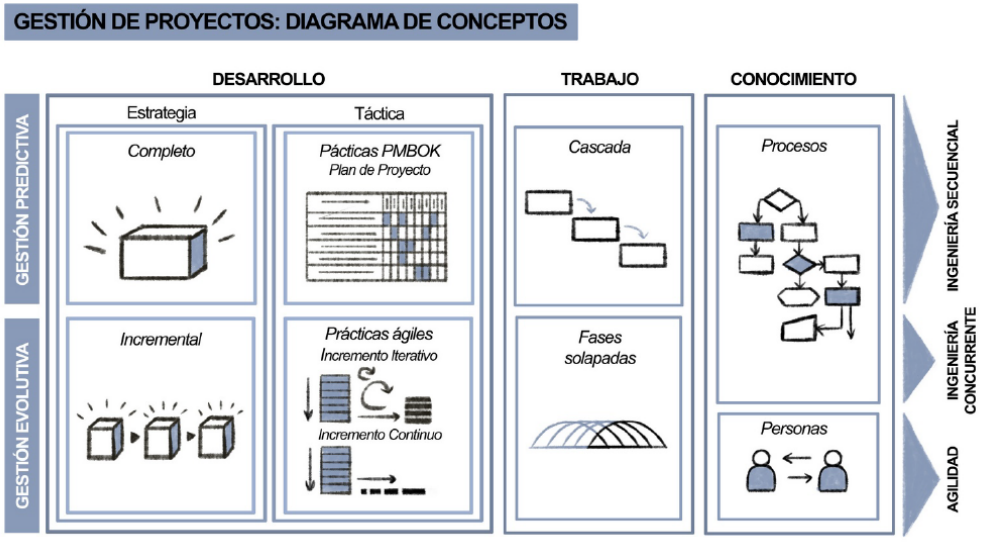


Figura 5. Características entre la gestión de proyectos predictiva vs evolutiva. Palacio (33).

3 http://agilemanifesto.org/iso/es/manifesto.html

4.2. ¿Qué es SCRUM?

Se explica en este apartado el marco teórico de Scrum, con el objetivo de explicar todo el marco de trabajo para que en el siguiente punto se presente la adaptación al PFG realizada.

4.2.1. Marco teórico.

SCRUM está basado en metodología ágil antes descrita. Los autores Ken Schwaber y Jeff Sutherland (36) lo definen de la siguiente forma:

"Scrum (n): Es un marco de trabajo a través del cual las personas pueden abordar problemas complejos adaptativos, a la vez que se entregan productos de forma eficiente y creativa con el máximo valor.

Scrum es ligero, fácil de entender y difícil de dominar. Scrum no es un proceso, una técnica, o método definitivo. Todo lo contrario, es un marco de trabajo donde se pueden emplear un conjunto de diferentes procesos y técnicas. Scrum muestra la eficacia relativa de las técnicas de gestión de producto y de trabajo de modo que podamos continuamente mejorar el producto, el equipo y el entorno de trabajo".

Palacio (33) lo define, con más concreción, como "*un modelo de desarrollo ágil caracterizado por:*

- *Equipos autónomos y autoorganizados que comparten su conocimiento de una forma abierta y aprenden juntos. De aquí el termino metafórico de "avanzar en Scrum".*
- *Una estrategia de desarrollo incremental, en lugar de la planificación completa del producto.*
- *Basar la calidad del resultado en el conocimiento tácito de las personas y su creatividad; no en la calidad de los procesos empleados.*
- *Solapar las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada*".

Toda esta teoría tiene su aplicación en una determinada forma de trabajar, básica, basada en artefactos o herramientas, en una serie de reuniones de carácter periódico y que se realizan a lo largo del proyecto y en unos roles que deben tomar los participantes del proyecto. Esta forma de trabajar hace de Scrum un marco o conjunto de buenas prácticas para la gestión de proyectos Sutherland (37). Este es el motivo de utilizar a veces el término metodología o marco de trabajo.

4.2.2. El ciclo Scrum.

Cuando se empieza a trabajar con Scrum es necesario leer documentación y seguir las instrucciones, es decir adoptar un marco estándar de trabajo para entender y aplicar las técnicas, reglas de aplicación, roles, artefactos y eventos. Menzinskyet al. (34), Palacio (35), Palacio (33) y Schwaber y Sutherland (36) establecen que los componentes del ciclo estándar de Scrum son:

- Roles:
    - Equipo de Desarrollo o Development Team.
    - Propietario del producto o Product Owner.

  - Scrum Master.
  - A los tres anteriores se les llama de forma conjunta "Equipo o Equipo Scrum o Scrum Team"
- Artefactos[4] o Artifacts:
  - Pila del producto o Product Backlog.
  - Pila del sprint o Sprint Backlog.
  - Incremento.
- Eventos:
  - Sprint o iteración.
  - Reunión de planificación del sprint o Sprint Planning.
  - Scrum diario o Daily Scrum.
  - Revisión del sprint o Sprint Review.
  - Retrospectiva del sprint o Sprint Retrospective.

Se inicia con la visión general del resultado que se desea y a partir de ella se especifica y da detalle a las funcionalidades que se desean obtener en primer lugar. Será el "sprint 0". En cada "ciclo de desarrollo o iteración o sprint" se finaliza con la entrega de una parte operativa (viable) del producto, es lo que se llama "incremento". El sprint es el evento clave de Scrum para mantener un ritmo de avance continuo, por tanto, es un periodo de tiempo acotado. La duración puede ser de una hasta seis semanas. El equipo monitoriza y revisa la evolución de cada sprint en reuniones breves diarias. En ellas se revisa el trabajo realizado por cada miembro el día anterior, y el previsto para el día actual. Se denominan "reunión de pie o scrum diario o stand-up meeting, daily scrum o morning rollcall".

La reunión de planificación del sprint marca el inicio de cada sprint, tomándose como base las prioridades y necesidades de negocio del cliente y se determinan cuáles y cómo van a ser las funcionalidades que se incorporarán al producto al terminar el sprint.

Al finalizar el sprint se repasa éste en las reuniones de revisión del sprint. De carácter informal, marcan el ritmo de construcción y la trayectoria que va tomando la visión del producto. En la reunión se debe ver y probar el incremento, obteniendo el propietario del producto y el equipo Scrum, un feedback relevante para revisar la pila del producto y a la misma vez identificar las tareas que están hechas y cuáles no.

Tras la finalización de la revisión del sprint corresponde realizar la reunión de retrospectiva del sprint, justo antes de la siguiente planificación de sprint. No estaba en el marco original de Scrum, pero se ha ido consolidando con el tiempo como un evento indispensable dentro del modelo. Normalmente es confundida con la reunión de revisión del

4 Según Palacio (33) y Sutherland (37) Los artefactos también se pueden llamar *herramientas*. Es una definición que gusta emplear porque son los elementos en los cuales el equipo Scrum se va a apoyar, de forma administrativa, para realizar las tareas encomendadas, es decir ayudan a los "roles" en la gestión de los "eventos".

sprint. El objetivo de la revisión del sprint es analizar "qué" se está construyendo: el producto; mientras que una reunión retrospectiva del sprint se centra en el marco de trabajo, es decir en el "cómo" se está construyendo. En el proyecto no se realizará la reunión de retrospectiva al tener el equipo un solo miembro.

Este ciclo iterativo hace que secuencialmente se vayan finalizando los sprints, revisando incremento, viendo que es el adecuado hasta la finalización de todos los sprints. En ese momento el proyecto puede cerrarse con el cliente y dar por finalizado la versión de ese producto. La visión en conjunto del ciclo estándar de Scrum se representa en Figura 6.

Palacio (33) afirma que en Scrum no existe ninguna autoridad que determine o dictamine lo que es Scrum y lo que no es. Está en constante cambio y seguirá evolucionando en el tiempo con las aportaciones de la comunidad de profesionales, como lo hizo desde sus inicios. De hecho, sus creadores Schwaber y Sutherland (36) lo definen alegando que no es un proceso, una técnica, o método definitivo. Sino todo lo contrario, es un marco de trabajo donde se pueden emplear un conjunto de diferentes procesos y técnicas.

4.3. Aplicación del ciclo Scrum al proyecto.

En la introducción al capítulo se han explicado los motivos de elección de esta metodología de gestión para el proyecto. Ahora en este apartado se explica la aplicación del ciclo Scrum al proyecto.

Al ser un proyecto académico no se realiza el estudio de factibilidad del proyecto. Es el primer documento necesario en un proyecto de cualquier tipo, ya que es el documento contractual, es decir un contrato legal, que se firmará con el cliente para la ejecución del proyecto requerido[5]. Evidentemente no forma parte de Scrum, sino de la lógica de negocio de cualquier empresa dedicada a la venta de servicios.

El siguiente paso es dotar al proyecto de una "visión del proyecto". Ésta es un reflejo de la investigación, estudio y producto a obtener. En el PFG se obtiene de la propuesta del proyecto aceptada y validada por Dirección de Proyectos. Es la siguiente: *Diseñar una estructura de datos y una aplicación Java que aplique Topic Modeling en un corpus de textos de software con el propósito de buscar términos relacionados (topics) y su relación con los documentos del corpus*.

Una vez la visión es definida, se da comienzo al ciclo Scrum y sus reglas, ilustradas en Figura 6. Como primera acción se debe establecer quiénes son los interesados en el proyecto y

5 Aunque este aspecto se puede equiparar a la matrícula de la asignatura que vincula a dos partes, universidad y alumno.

sus roles, que vienen definidos[6] en la Tabla 3. En dicha tabla se especifican diversos campos. Los más importantes son el rol en el proyecto y la información de evaluación.

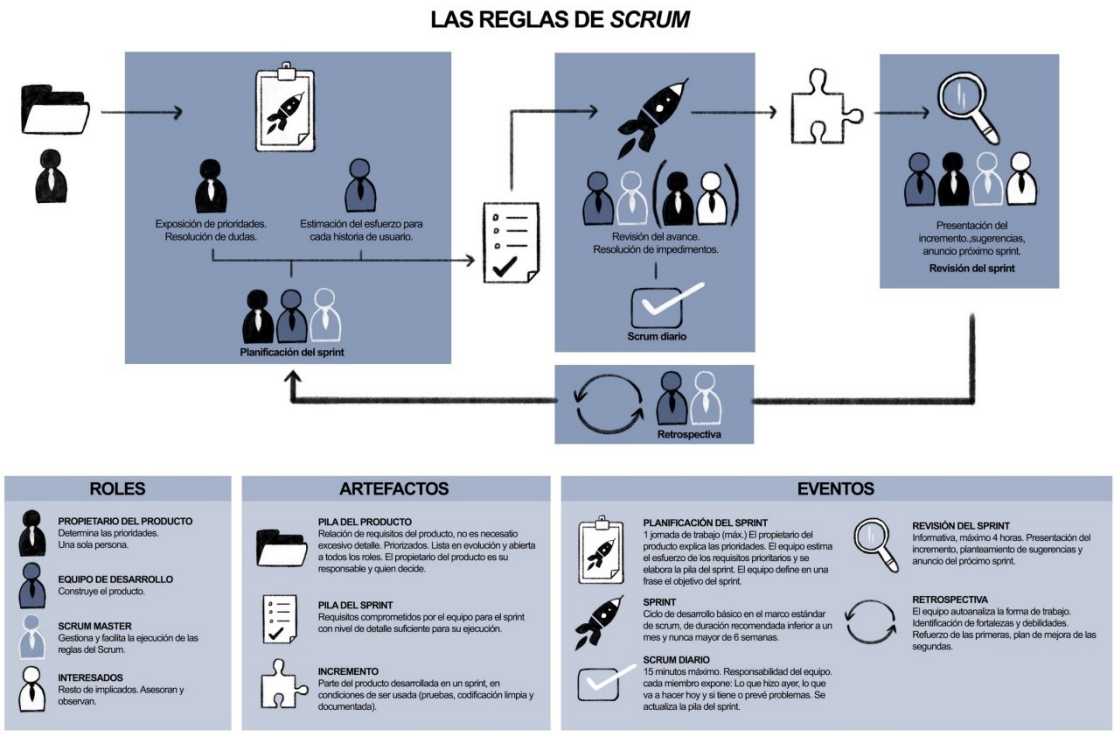


Figura 6: el ciclo Scrum. Palacio (33).

El alumno como la directora de proyecto asumen todos los roles del ciclo. Evidentemente el equipo de desarrollo es sólo una persona al ser un PFG unitario: el proyectando. La figura del cliente está asociada al propietario del producto y recae sobre la directora del proyecto y por tanto el equipo de Scrum o Scrum Team está formado por el alumno y directora de proyecto.

Tabla 3. Registro de interesados del PFG. Adaptado de PMOInformatica (38).

Registro de interesados (Stakeholders)

Información de identificación						Información de evaluación	
Nombre	Puesto	Organización / Empresa	Ubicación	Rol en el proyecto	Información de contacto	Requisitos principales	Expectativas principales
Valerio García Palao	Alumno	Universidad San Jorge	Yecla	Scrum Master	alu.53701@usj.es	Memoria	Asignatura PFG
				Equipo de Desarrollo o Development Team.		Aplicación Java	
						Defensa proyecto	
M. Francisca Pérez Pérez	Directora PFG	Universidad San Jorge	Zaragoza	Propietario del producto o Product Owner	mfperez@usj.es	Memoria	Dirección de proyecto
						Aplicación Java	Cliente
Tribunal	Tribunal de Proyectos	Universidad San Jorge	Zaragoza	Interesados	no aplica	Evaluación	PFG apto

Existen dos columnas, llamadas "requisitos principales" y "expectativas principales", que forman parte de la "información de evaluación". En la columna de requisitos, se describe que requiere del proyecto, en términos de entregables o información, esa persona. En la columna de expectativas se relacionan los beneficios que la persona espera obtener del proyecto.

6 Las tablas de registro de interesado, pila del producto y lista de tareas de la iteración, son adaptaciones realizadas de plantillas gratuitas estándar de PMOInformatica (38).

Seguidamente se presenta la Tabla 4 o "pila del producto inicial para el proyecto" o "Product Backlog". En ella se establecen una serie de funcionalidades y/o requerimientos y/o requisitos para el proyecto. Tras reunión inicial entre Product Owner y Scrum Master, se crean una serie de ítems en función de los objetivos del PFG.

Tabla 4: Pila del producto inicial o Initial Product Backlog del PFG.

Pila del Producto Inicial o Initial Product Backlog.

Identificador (ID)	Enunciado de la Historia/Requerimiento/Funcionalidad	Alias	Estado	Dimensión / Esfuerzo	Iteración (Sprint)	Prioridad /Orden
OS-08	Investigar y aprender sobre metodología SCRUM para aplicar al desarrollo del proyecto	Scrum	Planificada	21	0 y 1	1
OS-01	Investigar y estudiar el estado del arte desde Data Science a Topic Modeling para comprender la técnica a utilizar.	Estado del arte DS-TM	Planificada	34	0 al 2	2
OS-02	Establecer los fundamentos teóricos y conceptuales de Topic Modeling y LDA para su uso en la aplicación a desarrollar.	Fundamentos TM-LDA	Planificada	55	1 y 2	3
OS-03	Establecer la relación de Topic Modeling con el Análisis de Textos de Software para relacionar términos de software.	Relación TM-Análisis Software	Planificada	55	2 y 3	4
OS-04	Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.	Mallet	Planificada	34	3 y 4	5
OS-09	Crear infraestructura hardware y software para desarrollo de la aplicación y su evalución.	Infraestructura H+S	Planificada	5	3	6
OP-01	Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado en el proyecto	Diseño estructura datos	Planificada	21	4 y 5	7
OS-05	Analizar la estructura de datos que se reciben en la técnica para posteriormente aplicar Topic Modeling, muestreo y ver relaciones entre términos.	Análisis estructura datos	Planificada	13	4	8
OP-02	Implementar una aplicación en Java que utilice la técnica Topic Modeling para buscar términos relacionados	Implementar aplicación	Planificada	144	3 al 7	9
OS-06	Establecer escenarios en los que se implementará y analizará la aplicación Java para evaluar su comportamiento	Escenarios de aplicación	Planificada	8	7	10
OS-07	Evaluar comportamiento de Topic Modeling en el conjunto de datos del corpus seleccionado para ver resultados y establecer conclusiones	Evaluar	Planificada	21	8	11
OS-10	Realizar la memoria del PFG	Memoria	Planificada	233	2 al 9	12
OS-11	Demostrar y defender el PFG	Defensa	Planificada	8	9	13
OS-12	Tareas relacionadas con la administración y gestión de Scrum	Scrum administración	Planificada	13	0 al 9	---

TOTAL Dimensión/Esfuerzo = 665 puntos

En la tabla, se ha creado un identificador de producto único[7], un alias, un estado, que inicialmente estará en la condición de "planificada", y luego deberá ir cambiado a estados como "en proceso", "hecha", "pruebas" o incluso "descartada". Se aporta un valor de esfuerzo o dimensión del ítem. Esta medida, en este caso, se expresa en puntos[8]. También, se dota a la tabla de una columna de orden o prioridad. Ella está ordenada de mayor (valor=0) a menor (valor=13) según prioridad de las funcionalidades o requisitos. Como se ha planteado como proyecto, se ha añadido el requerimiento de realizar una administración y gestión del proyecto, que aunque no tiene prioridad, se cree necesario porque todo proyecto tiene un gasto de tiempo en estas tareas, incluso en Scrum. Para finalizar, se crea la columna iteración o sprint, indicando en cada celda el sprint o sprints en los cuales esa funcionalidad o requerimiento se

7 Identificador y descripción de la tarea se corresponden con los objetivos del proyecto, descritos en el capítulo 3.

8 Es aconsejable en Scrum expresar en el Product Backlog la dimensión del ítem en puntos y desligar la medida absoluta, por ejemplo, en horas, con una medida relativa como los puntos. Para la asignación de esfuerzo a una tarea, se suelen utilizar algunas técnicas que permiten unas estimaciones con el grupo de trabajo más acordes a la carga o esfuerzo de la tarea. La técnica más común es la de estimación de póquer, que tiene su base en la serie de Fibonacci. En el proyecto se utiliza esta técnica para la asignación de esfuerzo en puntos a la tarea.

realizará. Aunque en Scrum puro o teórico no haría falta nada más que calificar el primer sprint, al ser un PFG se presenta inicialmente una asignación de sprints para cada fila de la tabla. Esta asignación puede cambiar en cada iteración donde se hace la revisión de la pila de producto, según las prioridades indicadas por el propietario del producto.

La gestión de la pila de producto es fundamental en el proyecto, pues según lo introducido en la pila de producto así van a funcionar los sprints, diferenciando y detallando las tareas, con la obtención de nuevas partes del producto.

Scrum no necesita una planificación total previa, sino solamente una planificación inicial, desde dónde poder empezar a trabajar. Como se está realizando un PFG, se realiza la "planificación inicial de tareas" o "Sprint Backlog", en función de los ítems de la pila del producto de la Tabla 4, dando lugar a las diferentes iteraciones o sprints en la Tabla 5. En ella aparecen campos repetidos de la tabla anterior que identifican a los ítems. Estos no se han dividido en tareas u otras funcionalidades o requisitos y se realizarán más adelante, conforme el proyecto avance.

Aparecen nuevas columnas, siendo la primera, la creación de los diferentes sprints. Cada sprint corresponde con una determinada fase temporal. Se ha planificado un ritmo de 4 semanas por sprint, al ser el equipo de desarrollo de una persona y no necesitar reuniones a más corto plazo. Cada sprint deberá acabar con su reunión de revisión del sprint, retrospectiva del sprint y planificación del siguiente sprint. En cada columna, correspondiente al sprint, se ha incluido el esfuerzo que un determinado ítem tiene en ese periodo de tiempo. Cada sprint, en esta planificación inicial tiene un esfuerzo o dimensión similar. Si bien el primer sprint, por ser de planificación y arranque, junto con el último sprint, por ser el de presentación y defensa del proyecto, tienen menos esfuerzo que el resto. El sprint 0 contiene el ítem de arranque del proyecto en sí: planteamiento, matriculación y lanzamiento del proyecto, junto con los primeros ítems de trabajo y con el estado de finalización en "realizada" puesto que ese ítem ya está realizado cuando se construye la tabla.

En las reuniones de planificación del sprint, los ítems podrán descomponerse en historias de usuario[9] y/o tareas según se vea adecuado.

Cada sprint se identifica por su número, nombre, fechas de duración, número de días, horas estimadas de trabajo, y velocidad del equipo aproximada durante su duración, expresada en puntos por semana. También en dicha tabla se muestra la relación de horas y su desglose según Guía Docente del PFG 2019-20. Estas sirven de orientación para establecer el número máximo aproximado de horas por parte del equipo de desarrollo.

9 Aunque en el marco teórico de Scrum no se especifican o prescriben las historias de usuario, sino Sutherland (37) tareas a realizar, es habitual utilizarlas para documentar los elementos de la pila de producto o lista de iteraciones. Las historias de usuario resumen las características del producto y expresan lo que el cliente quiere que se implemente.

Tabla 5. Lista de tareas inicial de la iteración o Initial Sprint Backlog del PFG.

Lista de tareas inicial de la iteración o Initial Sprint Backlog

Identificador (ID) del Product Backlog	Enunciado del item de Product Backlog	Estado	SPRINT										CAPÍTULO ASOCIADO
			0	1	2	3	4	5	6	7	8	9	
OS-08	Investigar y aprender sobre metodología SCRUM para aplicar al desarrollo del proyecto	Planificada	8	13									4
OS-01	Investigar y estudiar el estado del arte desde Data Science a Topic Modeling para comprender la técnica a utilizar.	Planificada	10	11	13								2
OS-02	Establecer los fundamentos teóricos y conceptuales de Topic Modeling y LDA para su uso en la aplicación a desarrollar.	Planificada		34	21								2
OS-03	Establecer la relación de Topic Modeling con el Análisis de Textos de Software para relacionar términos de software.	Planificada			34	21							2
OS-04	Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.	Planificada				21	13						5
OS-09	Crear infraestructura hardware y software para desarrollo de la aplicación y su evaluación.	Planificada				5							5
OP-01	Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado en el proyecto	Planificada					13	8					5
OS-05	Analizar la estructura de datos que se reciben en la técnica para posteriormente aplicar Topic Modeling, muestreo y ver relaciones entre términos.	Planificada					13						5
OP-02	Implementar una aplicación en Java que utilice la técnica Topic Modeling para buscar términos relacionados	Planificada				21	34	34	34	21			5
OS-06	Establecer escenarios en los que se implementará y analizará la aplicación Java para evaluar su comportamiento	Planificada								8			5
OS-07	Evaluar comportamiento de Topic Modeling en el conjunto de datos del corpus seleccionado para ver resultados y establecer conclusiones	Planificada									21		7
OS-10	Realizar la memoria del PFG	Planificada			21	13	5	34	50	55	50	5	Memoria
OS-11	Demostrar y defender el PFG	Planificada										8	Defensa
OS-12	Planteamiento, matriculación y lanzamiento	Realizada	3	1	1	1	1	1	1	1	1	2	Admon.

De:		09-sep-19	30-sep-19	28-oct-19	25-nov-19	07-ene-20	03-feb-20	02-mar-20	30-mar-20	27-abr-20	25-may-20		
A:		27-sep-19	25-oct-19	22-nov-19	20-dic-19	31-ene-20	28-feb-20	27-mar-20	24-abr-20	22-may-20	19-jun-20		
Nombre sprint:		Lanzamiento	Estudio I	Estudio II	TM y AS (I)	TM y AS (y II)	Aplicación (I)	Memoria (I)	Aplicación (yII)	Memoria (yII)	Presentación		
Release o condiciones de satisfacción:		Matriculación	Estado arte	Estado arte	Relación TM-AS	TM y MALLET	Aplicación	Capítulos	Plug-in	Capítulos	Defensa PFG		
Dimensión / Esfuerzo:	665	21	59	90	82	79	77	85	85	72	15		
Velocidad del equipo (puntos/semana):	17,05	7,00	14,75	22,50	20,50	19,75	19,25	21,25	21,25	18,00	3,75		
Número de semanas =	39,00	3	4	4	4	4	4	4	4	4	4		
Número de días =	242,00	18	25	25	25	24	25	25	25	25	25		
Número de horas estimadas =	284,98	21	29	29	29	28	29	29	29	29	29		
Número de horas estimadas TOTALES según Guía Docente =	300,00												
Número de horas estimadas trabajo personal													
Horas según Guía Docente =	285,00												
Reuniones de seguimiento con el Director/Tutor (10 horas máximo =	10,00												
Actividades de evaluación =	1,00												
Otras actividades prácticas =	4,00												

5. DISEÑO Y DESARROLLO.

En el capítulo 4 se describe la presentación de la metodología ágil Scrum y cómo se ha adaptado para la realización en un proyecto de carácter académico. En el apartado 4.2 se comenta el ciclo Scrum, con sus roles, artefactos y eventos, que se van a utilizar para el desarrollo del proyecto y en el apartado 4.3 se explica esa adaptación. En dicho apartado se incorpora la Tabla 5 o *Sprint Backlog Inicial* con la configuración temporal de los objetivos a realizar en este proyecto. De ella, se obtienen los objetivos que pertenecen al diseño y desarrollo, la prioridad y asociación a los sprints en los cuales se desarrollarán. En el capítulo 7 se incorpora la Tabla 18 o *Sprint Backlog Final* que muestra cómo definitivamente se ha ejecutado el proyecto y como quedan los objetivos de diseño y desarrollo. En la Figura 6, en forma de línea temporal, se muestran dichos objetivos, orden y asociación a los sprints.

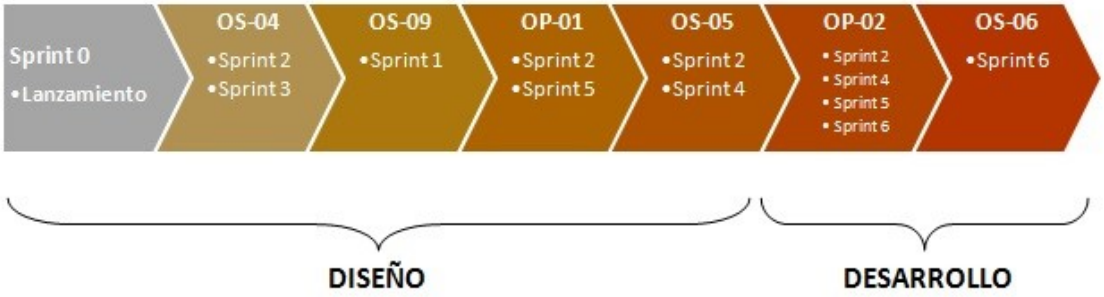


Figura 7: línea temporal del capítulo 5, basada en Tabla 18 del capítulo 7.

Cada cheurón o bloque de la figura, representa una sección del capítulo, donde se describe el diseño y desarrollo para alcanzar el objetivo, y uno o varios sprints en los que se aborda en forma de subsección. La única excepción la tiene el sprint 0 que, como arranque del proyecto, se presenta como una sección y no como una subsección de un capítulo.

5.1. Sprint 0: lanzamiento del proyecto.

Conforme a lo explicado en la introducción del capítulo, esta sección aborda el sprint 0-lanzamiento en su totalidad.

Scrum no necesita de una planificación total previa, sino una planificación inicial, guiando que el primer sprint, en este caso el número cero, sea el arranque o inicio del proyecto en sí. Se detallan en la Tabla 6 las acciones realizadas en este sprint, conteniendo el proceso de matriculación e inicio de la gestión del proyecto. Además, se le dota de dos ítems[10] de trabajo (OS-08 y OS-01) dando así forma a esta tabla y convirtiéndose en un contenedor de requisitos, requerimientos y funcionalidades de todas las necesidades del cliente y equipo de desarrollo.

10 Los ítems que se referencian en este capítulo son los códigos ID de las tablas de Product Backlog y Sprint Backlog. En este caso, estos dos objetivos están en el mismo sprint en la Tabla 5 o Tabla 18 al ser el inicio del trabajo.

Tabla 6: Sprint 0 - lanzamiento. Objetivos, requisitos, requerimientos y funcionalidades.

SPRINT ID	SPRINT DESCRIPCIÓN	ESF.	OBJ ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES ID	DESCRIPCIÓN	TIPO
0	Lanz.+Est (I)	39				
			OS-08	*Investigar y aprender sobre metodología SCRUM para aplicar al desarrollo del proyecto*		
				8,00	Investigar sobre Scrum y metodolgías ágiles	Organización
				8,01	Leer literatura seleccionada sobre Scrum (I)	Organización
				8,02	Leer literatura seleccionada sobre Scrum (y II)	Organización
				8,03	Análisis sobre software open/free de Scrum para usarlo como herramienta	Organización
				8,04	Carga de datos en software seleccionado	Organización
				8,05	Repaso de conceptos para capítulo 4	Organización
			OS-01	*Investigar y estudiar el estado del arte desde Data Science a Topic Modeling para comprender l*		
				1,90	Leer literatura seleccionada sobre Data Science a Topic Modeling (i)	Organización
			OS-12	*Tareas relacionadas con la administración y gestión de Scrum*		
				12,00	Listado propuestas PFG	Organización
				12,01	Elección y solicitud de PFG	Organización
				12,02	Asignación del PFG	Organización
				12,03	Leer guía docente del PFG	Organización
				12,04	Leer manual del PFG	Organización
				12,05	Leer normativa interna sobre propiedad intelectual de los trabajos en USJ	Organización
				12,06	Reunión revisión sprint 0 y planificación sprint 1	Organización

Paralelamente a la parte administrativa, se empieza a trabajar con los dos ítems de trabajo señalados en la columna 0 de la Tabla 5:

- OS-08, asociado a la metodología Scrum y que por tanto se desarrolla completamente asociado al capítulo de metodología.
- OS-01, investigar y estudiar el estado del arte que va desde Data Science a Topic Modeling, con el objetivo de comprender la técnica a utilizar. Se selecciona una extensa bibliografía y se comienza su estudio. La síntesis de este estudio se plasma en el capítulo 2.

Para la reunión de revisión del sprint, se desarrolla y ajusta la planificación inicial, Tabla 5 del capítulo 4, a la mostrada en Tabla 18 del capítulo 7. Es un ajuste fuerte y se debe a una nueva planificación de trabajos para los sprints 0 y 1, dando lugar a una ampliación de la fecha de revisión de estos dos sprints y afectando a la planificación del resto de sprints. El cambio de planificación se origina por la necesidad de mayor tiempo para OS-01 y OS-08 y por tanto no pueden comenzar a desarrollarse lo objetivos siguientes.

5.2. OS-04: Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.

En la Tabla 7 se enumeran los detalles de trabajo realizados en este objetivo.

5.2.1. Sprint 2.

En este sprint se parte de la literatura seleccionada sobre Topic Modeling (TM) en OS-01 (capítulo 2) y se realiza un proceso de selección sobre la misma con el objetivo de conseguir bibliografía que presente herramientas para uso en TM, analizar las mismas y seleccionar una de ellas para su uso en la aplicación.

Se detecta una amplia bibliografía, sobre todo desde principio del siglo, y concretamente desde el año 2007, fecha en la que Linstead et al. (19) dan como comienzo de

la aplicación de este tipo de métodos de aprendizaje automático al análisis de código de software[11]. También en capítulo 2 se muestra la Figura 3 dónde Agrawal et al. (17) relacionan el uso de TM para el análisis de software.

Tabla 7: OS-04. Sprint y RyR asociados.

OBJETIVO ID	OBJETIVO DESCRIPCIÓN	SPRINT ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES DESCRIPCIÓN	TIPO
OS-04	Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.				
		2	4,00	Repaso literatura seleccionada sobre Topic Modeling obtenida de OS-01	Diseño
			4,01	Seleccionar literatura de TM que contenga ejemplos de herramientas	Diseño
			4,02	Filtrar herramientas sobre herramientas vistas y seleccionar una	Diseño
		3	4,10	Leer documentación general sobre la herramienta	Diseño
			4,11	Leer documentación específica sobre API o librerías que contenga	Diseño
			4,12	Pruebas con la herramienta seleccionada para determinar uso en TM	Diseño
			4,13	Pruebas específicas con la herramienta seleccionada para determinar uso en TM asociado al Análisis de Software. Ver uso de API o librerías en lenguaje Java	Diseño
			4,14	Documentar pruebas para presentar a Dirección Proyecto	Diseño
			26,10	Estudiar ejemplos de procesamiento para aprender uso de la herramienta en línea de comandos	Diseño
			26,11	Estudiar ejemplos de ficheros en la aplicación de línea de comandos	Diseño

Se presenta la Tabla 8, como un resumen de las principales herramientas que se han consultado o investigado. Se incluye la referencia bibliográfica de dónde se obtiene el uso de la herramienta en ese artículo o libro, nombre de la herramienta, tipo y uso que se le da en el artículo. Se añade además tres recursos bibliográficos dónde aparecen más herramientas de las propuestas en la tabla para el procesamiento de lenguaje natural, minería de textos, topic modeling, etc. Se señala con un tono gris las que interesan para el proyecto, es decir las que emplean la herramienta para Text Mining o Topic Modeling. La segunda clave de búsqueda o de selección, es que sean librerías y adaptadas o construidas en JAVA. El tercer criterio de selección ha sido que fueran con licencias de acceso y uso libre.

López-Nozal et al. (20) presentan una tabla comparativa en su estudio sobre las diferentes herramientas analizadas por ellos y sus características, viendo en las aplicaciones el algoritmo base empleado en ellas, destacando LDA sin modificaciones. Bhatia y Bostjan (9) tienen un estudio más completo sobre las herramientas estudiadas, seleccionando para la zona de minería de textos y concretamente para Topic Modeling y detección de spam a MALLET (39). Se toma como referencia y se comprueba que aparece en referencias de otros autores como López-Nozal et al. (20), Thomas et al. (40) o Rehurek y Sojka, utilizándose para el análisis de software con Topic Modeling. En Burton (41) habla sobre MALLET, diciendo que tiene implementado Topic Modeling con LDA y otros algoritmos. Se comprueba que esta herramienta es muy utilizada en el análisis de textos con Topic Modeling en las Humanidades Digitales, referenciada en autores Silvestre (16), Chandía (42) o Calvo (43).

11 Autores como López-Nozal et al. (20) comentan que en 2004 en la creación de la Conferencia Internacional Mining Software Repositories (MSR http://www.msrconf.org/) aparece la necesidad de analizar los datos disponibles en los repositorios de software y por tanto puede tomarse como una fecha de inicio de aplicación de técnicas al análisis de software, debido a qué si hay repositorios hay necesidad de explotación de los datos del mismo y por tanto hay necesidad de aplicación de herramientas para el modelado de la información.

Por tanto, tras estudiar la técnica y grupo de herramientas, se decide utilizar la herramienta MALLET (39), como librería para desarrollar la aplicación requerida en el objetivo OP-02. Cumple esta herramienta con los tres requisitos comentados anteriormente en el análisis de la tabla de herramientas y es utilizado para Topic Modeling en el análisis de software por diferentes autores como se ha visto; teniendo también una implementación bastante eficiente de la serie de algoritmos LDA.

Tabla 8: resumen de herramientas consultadas para Topic Modeling.

BIBLIOGRAFÍA	HERRAMIENTA	TIPO	USO
Guantiva et al.	Yate (Yet another Term Extractor)	Extracción de candidatos a términos nominales (CAT)	Text mining
Bhatia & Bostjan	Waikato Environment for Knowledge Analysis (WEKA)	Librería	General ML
Bhatia & Bostjan	Java Machine Learning Library (Java-ML)	Librería	General ML
Bhatia & Bostjan	Apache Mahout project	Librería	Clasificación, clustering
Bhatia & Bostjan	Apache Spark,	Plataforma tratamiento datos	General ML
Bhatia & Bostjan	Deeplearning4j (DL4J)	Librería	Deep learning
Bhatia & Bostjan	Machine Learning for Language Toolkit (MALLET)	Librería y aplicación línea comandos	Text mining
Bhatia & Bostjan	Encog Machine Learning Framework	Framework	General ML
Bhatia & Bostjan	ELKI: Environment for Developing KDD-Applications Supported by Index-Structures	Framework	Data mining
Bhatia & Bostjan	Massive Online Analysis (MOA)	Framework	General ML
López-Nozal et al.	Machine Learning for Language Toolkit (MALLET)	Librería y aplicación línea comandos	Text mining
López-Nozal et al.	A Java Implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling for Parameter Estimation and Inference (JGibbLDA)	Librería	Text mining
López-Nozal et al.	Embedded Topic Extraction (EmbTE)	Aplicación	Text mining
López-Nozal et al.	Software Defect Clustering (SDCL)	Aplicación	Text mining
Saedi et al.	ITMViz: Interactive Topic Modeling for Source Code Analysis	Aplicación	Topic Modeling
Wang & Liu	Java Software Engineers Assistant (JSEA)	Aplicación (MALLET incorporado)	Topic Modeling
Thomas et al.	Machine Learning for Language Toolkit (MALLET)	Librería y aplicación línea comandos	Text mining
Gretarsson et al.	TopicNets: Visual Analysis of Large Text Corpora with Topic Mod	Aplicación	Topic Modeling
Asunción et al.	Topically-Rich Artifact Search Engine Toolkit (TRASE toolkit)	Suite de aplicaciones	Topic Modeling
Bissyandé et al.	Orion: software project engine	Arquitectura para búsquedas	Text mining
Rehurek & Sojka	OpenNLP	Librería	NLP
Rehurek & Sojka	NLTK: The Natural Languaje Toolkit	Librería	NLP
Rehurek & Sojka	Machine Learning for Language Toolkit (MALLET)	Librería y aplicación línea comandos	Text mining
Rehurek & Sojka	GATE: a General Architecture for Text Engineering	Arquitectura para text mining	Text mining
Wang et al.	The Peacock system	Aplicación	Topic Modeling
Newman et al.	Improving Search and Discovery Using Topic Modeling	Aplicación	Topic Modeling
Chandía	NLTK: The Natural Languaje Toolkit	Librería	NLP
Otros recursos consultados.			
https://radimrehurek.	GENSIM: topic modeling for humans	Librería	Topic Modeling
https://corpus-analys	Tools for Corpus Linguistics	Informe de aplicaciones	Data mining
https://en.wikipedia.(	Wikipedia: Outline of natural language processing	Informe sobre NLP y aplicaciones	NLP

5.2.2. Sprint 3.

Este sprint, según especifica la Tabla 7, contiene acciones para familiarizarse con la herramienta elegida: MALLET; viendo en profundidad todo el paquete de librerías que contiene y la realización de diferentes pruebas, para comprobar que sirve al objetivo principal del proyecto. Se divide en diferentes apartados que contemplan el desarrollo de los requisitos.

¿Qué es MALLET?

Según McCallum (39), en la web de MALLET: "*es un paquete basado en Java para el procesamiento estadístico de lenguaje natural, clasificación de documentos, agrupamiento, topic modeling (modelado de temas), extracción de información y otras aplicaciones de*

aprendizaje automático de textos. Además de la clasificación, MALLET incluye herramientas para el 'secuence tagging' para aplicaciones como la extracción de entidades con nombre del texto…Los 'topic models' son útiles para analizar grandes colecciones de texto sin etiquetar. El kit de herramientas de 'topic modeling' MALLET contiene implementaciones eficientes y basadas en muestreo de la asignación de Dirichlet latente (LDA), la asignación de Pachinko y la LDA jerárquica[12]*…Incluye rutinas para transformar documentos de texto en representaciones numéricas que luego pueden procesarse de manera eficiente. Este proceso se implementa a través de un sistema flexible de pipes (tuberías, que maneja tareas distintas, como tokenizar cadenas, eliminar palabras vacías y convertir secuencias en vectores de conteo*".

Por tanto, la calificación de herramienta para Machine Learning es correcta y está abarcando en su uso la aplicación tanto para NLP, como clasificación de documentos, clustering, topic modeling y extracción de información. Además, cuenta con dos formas diferentes de trabajo: un programa ejecutable sobre la línea de comandos y una API para su integración en otro software. Esta doble forma de uso hizo otro aspecto más a favor de la elección de la herramienta para este proyecto.

API de MALLET.

Su función general es doble y sirve para ratificar más su uso como herramienta en el proyecto. Por una parte, una mayor flexibilidad que la ofrecida por otras herramientas de importación y tratamiento de datos, tanto a nivel de API como de línea de comandos. Y por otra, la amplia variedad de librerías de las que consta para realizar diversas tareas.

Se examina en profundidad todo el compendio de paquetes y clases que componen la API en sí. Una vez realizado el estudio, se determina que paquetes tendrán mayor uso, en función de las clases vistas en ellos y serán "*topics*", "*types*" y "*pipe*", conformado así la primera serie de herramientas que se pueden emplear en el proceso de Topic Modeling.

¿Cómo hacer Topic Modeling?

Como se comenta en la Tabla 7, se determina realizar una serie de ejemplos como entrenamiento del uso de la herramienta y comprobación de su aplicación al objetivo del proyecto: analizar texto de código de software y buscar términos relacionados. En este apartado se explica brevemente cómo implementar Topic Modeling y posteriormente se comentan las pruebas realizadas.

Básicamente se necesita para realizar Topic Modeling:

12 Para una visión completa de los algoritmos implementados por MALLET en su totalidad puede visitarse https://www.programcreek.com/java-api-examples/index.php?project_name=mimno%2FMallet# en su fichero "*readme.md*".

a) *Un corpus sobre el que realizar el proceso de modelado*. Como son herramientas de aprendizaje automático en su base, cuanto más amplio sea el corpus mejor se comportará el modelo.
b) *Software que implemente el modelo y basado en un algoritmo*. Ver Tabla 8 para diferentes ejemplos. En este proyecto se ha elegido MALLET y sus herramientas para realizar Topic Modeling.
c) *Un proceso de visualización de los datos*. Será la forma de entender lo que el programa dice y hacer uso de la información generada.

Pero esto que es tan trivial tiene una complicación inherente a la hora de la implementación. De hecho, el estudio de la bibliografía sobre Machine Learning, Text Mining y Topic Todeling, arroja que el punto "b" es bastante complicado, sobre todo porque hay que realizar procesos de extracción y depuración de la información antes de modelar. Bathia y Kaluza (9) proponen un proceso de extracción de la información descrito en la Figura 8.

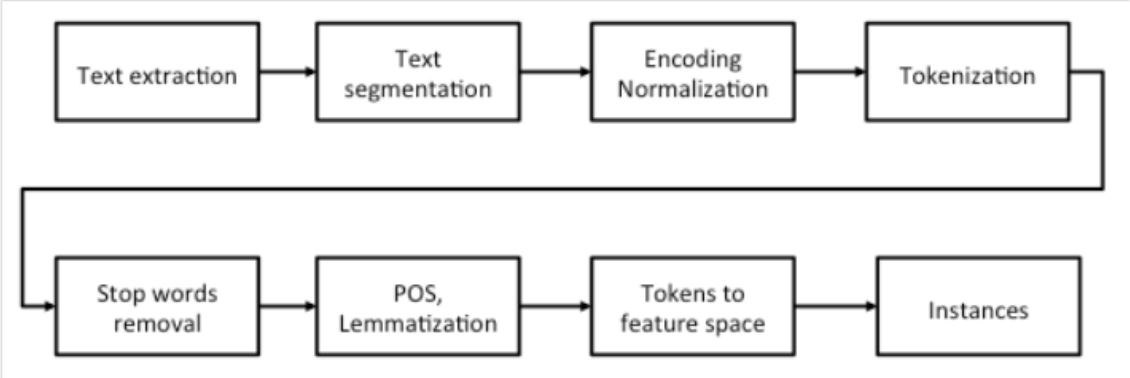


Figura 8. Proceso de extracción de información de lenguaje natural no estructurado. Bathia y Kaluza (9).

El proceso abarca el reto de transformar lenguaje natural no estructurado en estructuras basadas en instancias que contengan información estructurada.

Primero, se extrae el texto de una fuente, bien sea de Internet, documentos existentes o bases de datos. Es decir, aparece la primera fase que sería la importación de datos del corpus sobre el que trabajar. MALLET obliga que sea siempre en texto plano y el fichero acabado en ".TXT", aunque se comprueba que tiene otras opciones para interiorizar otros formatos a "TXT" y por tanto trabajar con ficheros de texto, pero extensiones diferentes. Así puede ya contemplarse ficheros de tipo JAVA y/o tipo "txt".

En el siguiente paso, que se suele llamar "*pre-procesado de los datos*", consta de varios pasos. Se comienza extrayendo el texto real y segmentarlo en partes del documento, por ejemplo, título, etiqueta, resumen y cuerpo. El tercer paso es la "*normalización*". Como depende del lenguaje natural empleado, así como la codificación utilizada en la escritura ofimática, por

ejemplo, Unicode UTF-8; es importante para asegurar que los caracteres se presenten de la misma manera. Para MALLET, se selecciona UTF-8 para los ejemplos.

A continuación, la "*tokenización*" divide el documento en palabras y genera los llamados "*tokens*". Es una representación de un fragmento de texto, generalmente una sola palabra, al que se pueden adjuntar propiedades. El siguiente paso elimina las palabras frecuentes que usualmente tienen una baja capacidad predictiva, es decir, lo que se llama *eliminación de "stopwords"* o palabras usuales del idioma: artículos, adverbios, pronombres, etc.

Dentro del pre-procesado, el último paso es el que se conoce como "*P.O.S. (Part-Of-Speech)"* dónde se genera un proceso para etiquetar y "*lematizar*" las palabras de cada "*token*" a su forma básica. Ésta se conoce como lema, y trata de eliminar las terminaciones de las palabras y sus modificadores. Por ejemplo, "corriendo" se transforma a "correr". El enfoque se basa en aislar las palabras del contexto en el que operen.

Ahora que se tienen *"tokens"* se trata de realizar el "*modelado*" o el último paso, que transforma "*tokens*" en lo que se suele llamar un "*espacio de características*". Realmente este espacio es conocido como "*Bag-Of-Words" (BoW),* es decir, es una bolsa de palabras que refleja todas las palabras que aparecen el texto y la frecuencia o cantidad de veces que aparece en el documento. Así, cada documento se presenta como un vector, que cuenta la cantidad de veces que cada palabra en particular aparece en el documento. Tales vectores finalmente se convierten en "*instancias*" para una mejor manipulación, por tanto, en MALLET se representan los datos en base a instancias. Éstas siempre incluyen un objeto de datos, pero también pueden incluir un nombre, en contextos de clasificación una etiqueta o una ruta al fichero.

Finalmente, las instancias son modeladas aplicando LDA y obteniendo los "*topics*" o "*temas*". Este proceso, por ejemplo, en MALLET, se realiza con la clase "ParallelTopicModel" u otros de los algoritmos que tiene implementados: LDA, DMR LDA, Hierarchical LDA, etc.

La parte de visualización es la parte de interacción directa con el cliente final y debe contemplar la entrada de datos e informar de los resultados, por ejemplo: topics, tokens asociados a cada Topic, matrices o vectores conteniendo probabilidades, alfabeto de palabras, etc. Existen muchas herramientas de visualización para los resultados, pero para el proyecto se realizará una parte de visualización propia, basándose en los requerimientos, requisitos y funcionalidades que el Product Owner describa.

Pruebas iniciales realizadas.

Se realizan varias pruebas con las herramientas de MALLET:

a) Prueba de su programa ejecutable en la línea de comandos llamado "mallet". Se utiliza el proceso de descrito por Raj (44) y se usa como corpus los ficheros de ejemplo obtenidos bien del directorio de ejemplos que lleva el paquete de MALLET o de los

corpus disponibles en Internet. Este ejemplo usa un fichero[13] obtenido de la web de MALLET llamado "ap.txt".

b) Ídem del anterior, pero reduciendo el fichero "ap.txt" a un contenido máximo de 3 ó 4 líneas para hacer pruebas rápidas.
    a. Prueba de un programa realizado en JAVA obtenido de la web de MALLET utilizándose el mismo corpus que en la prueba "a".
c) Ídem de "a", pero cambiando el corpus y las siguientes condiciones:
    a. Realizar una importación desde un directorio, con un par de ficheros dentro.
    b. Los ficheros no acaban en "TXT", sino acabados en "JAVA".
    c. Realizar un entrenamiento con pocos topics y pocas iteraciones para ver el funcionamiento.
d) Prueba de un programa realizado en JAVA obtenido de Bathia y Kaluza (9) y con el proceso descrito por ellos. Se utiliza para trabajar en este caso un corpus real[14].

En todas las pruebas los parámetros de modelización, como número de topics, número de iteraciones o parámetros alfa y beta, no cambian para que los valores de resultados no se vean afectados.

Todas las pruebas son consideradas como válidas para la realización de Topic Modeling, sin tener en cuenta el algoritmo LDA empleado. Concretamente la prueba "d" es considerada válida para ajustes en el diseño y desarrollo de la aplicación referida en el objetivo OP-02. La Figura 9 muestra una parte del resultado, concretamente los dos últimos topics encontrados.

```
array longtargetcounts[targetindex runnables smoothed weight topicsortedwords
8       0,5     betasum topic denseindex sample numtopics newtopic int index
alpha localtopicindex[denseindex protected numiterations nonzerotopics
initialize score threshold currenttypetopiccounts private topictermmass
burnin
9       0,5     public numtopics position topic import alphasum treeset
booleanout.println labelsequence tokens print system.currenttimemillis
totaltokens out.close save builder.append("\t printdocumenttopics info hours

<50> LL/token: -7,34442

Total time: 3 seconds
C:\mallet\TEST06~1>
```

Figura 9: Resultados de la ejecución de Topic Modeling sobre un corpus de ficheros Java.

La ejecución arroja que en 3 segundos se han analizado dos ficheros con más de 4.000 tokens y obteniendo, con 50 iteraciones, una lista de 10 topics relacionados. La lectura de los topics arroja claridad en el tema o temas a buscar: relación con Topic Modeling y sólo con dos ficheros seleccionados con código fuente relacionado con la técnica. A simple vista, se detecta

13 Disponible en: http://mallet.cs.umass.edu/topics-devel.php

14 Este corpus desarrollado por Greene & Cunningham, sobre más de 2000 noticias extraídas de la web de la cadena BBC, se utiliza como "benchmark" estándar en el estudio de "machine learning". Disponible en el sitio web: http://mlg.ucd.ie/datasets/bbc.html

una mejora en el proceso de importación, concretamente en la zona de obtención de token para que sean más concretos y así mejorar la generación de topics con más tokens relacionados. Ejemplo: eliminar palabras reservadas de Java.

5.3. OS-09: Crear infraestructura hardware y software para desarrollo de la aplicación y su evaluación.

En la Tabla 9 se enumeran los detalles de trabajo realizados en este objetivo, asociados al sprint 1.

Tabla 9: OS-09. Sprint y RyR asociados.

OBJETIVO		SPRINT	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES		
ID	DESCRIPCIÓN	ID	ID	DESCRIPCIÓN	TIPO
OS-09	Crear infraestructura hardware y software para desarrollo de la aplicación y su evaluación.				
		1			
			9,00	Preparar ordenador portátil (32 bits)	Sistemas
			9,01	Preparar ordenador auxiliar (64 bits)	Sistemas
			9,02	El IDE a utilizar será Eclipse	Sistemas
			9,03	Instalación de IDE Eclipse. Versión para portátil y ordenador auxiliar	Sistemas
			9,04	No se utilizarán bb.dd. A menos que sea necesario	Sistemas
			9,05	Actualizaciones de Eclipse en 32 y 64 bits	Sistemas

5.3.1. Sprint 1.

Se crea la infraestructura requerida para el funcionamiento de las herramientas y la aplicación generada. En un principio tal y como se comenta en el capítulo 6 sólo se plantea el uso de un ordenador portátil. Aunque más tarde se decide preparar una máquina virtual auxiliar con 64 bits como backup de desarrollo o máquina secundaria para casos de avería.

Restricciones.

En este objetivo es dónde aparecen restricciones que pueden determinar el avance en el desarrollo del proyecto y por tanto el desempeño de los objetivos. Se califican las siguientes:

1. De carácter temporal.
    a. Como aparece en el capítulo 4, existe una restricción asociada al tiempo de desarrollo del proyecto en el número de horas establecidas en el sprint backlog.
2. De carácter físico.
    a. Aparece una restricción asociada a la infraestructura hardware como se ha visto en el párrafo anterior y solventado con una máquina de backup. No obstante, en el manual de usuario se especifican las características técnicas básicas para el funcionamiento.
3. De carácter lógico.
    a. Restricción asociada al entorno de desarrollo. Debe ser Eclipse™ e instalado en las versiones adecuadas para los entornos de hardware seleccionados y sistemas operativos asociados.
    b. Restricción asociada al uso de base de datos. No se permite su uso.

c. Restricción asociada al tipo de fichero de entrada de datos: con código fuente escrito en lenguaje JAVA™ o texto plano, ambos en formato UTF-8.
d. Restricción asociada al texto de lenguaje asociado al análisis: el lenguaje elegido es JAVA™.
e. Restricción asociada a la aplicación a desarrollar: será un plug-in del IDE seleccionado, pero tendrá una clase "main" para poder ejecutar sin necesidad de la carga del plug-in.

5.4. OP-01: Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado en el proyecto.

Esta sección contempla el primer objetivo principal (OP-01) del proyecto, mostrando en la Tabla 10 los detalles de trabajo realizados en este objetivo, asociados a los sprints 2 y 5.

5.4.1. Sprint 2.

Este sprint, para este objetivo, comienza con la obtención de los requerimientos, requisitos y funcionalidades del Product Owner. De dicha reunión inicial, se obtienen, los que dan lugar a parte de las restricciones lógicas vistas anteriormente y la mayor parte de los que se presentan en el objetivo OP-02. Posteriormente al hito de reunión se diseñan las clases para la estructura de datos de entrada elegidas y para su conversión en datos útiles para Topic Modeling. En el proceso de diseño se vuelven a estudiar las estructuras de datos de JAVA para encontrar la más adecuada.

Tabla 10: OP-01. Sprint y RyR asociados.

OBJETIVO ID	OBJETIVO DESCRIPCIÓN	SPRINT ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES DESCRIPCIÓN	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES TIPO
OP-01	Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado en el proyecto				
		2			
			1,00	Obtener de Product Owner los requerimientos y funcionalidades de la aplicación para diseñar la arquitectura de datos	Organización
			1,02	Repasar estructuras de datos en Java	Diseño
			1,03	Diseñar las clases para las estructuras de datos elegidas.	Diseño
			1,04	Diseñar las clases para la conversión de estructura de datos de entrada a estructura de datos para Topic Modeling	Diseño
		5			
			1,05	Diseñar la arquitectura MVC para abordar el problema	Diseño
			1,06	Diseñar las clases para la vista de la aplicación	Diseño
			1,07	Diseñar las clases para el control de la aplicación.	Diseño

Datos de entrada. Diseño de clases.

De la reunión inicial se define el dato de entrada sobre el cual se planteará todo el aplicativo: análisis de ficheros de texto en formato UTF-8. El contenido puede ser código de software escrito en JAVA (fichero con extensión ".java") u otro tipo de texto (fichero con extensión ".txt"), es decir, el corpus es un conjunto de ficheros, alojados en uno o varios directorios, que son analizados y descartando los que no tengan la extensión especificada.

En el transcurso de la reunión también se incorpora otra opción de datos de entrada. Se trata de recibir datos por vía de otra estructura de datos: tipo tabla hash o tabla de dispersión. En este caso se recibe bastante depurado el corpus, estando compuesto de una lista de documentos, representados por objetos hash, con el binomio clave-valor, dónde la clave es el token y valor el número de veces que el documento contiene el token. Por tanto, la aplicación permite recibir uno o varios documentos de este tipo mediante las listas de hash.

La Figura 10 presenta el diagrama de clases diseñado para una estructura de datos, que pueda contener datos de los diferentes tipos especificados. En ella se distinguen dos paquetes: el paquete "usj.svit.topicmodeling.ui.corpus" y "usj.svit.topicmodeling.ui.tm".

El primer paquete se diseña para contener dos clases: "CorpusExamplesIT" y "CorpusExamplesSVIT". Cada una de ellas contiene ejemplos de datos que conforman los dos corpus sobre los que se trabaja: corpus de ficheros Java y corpus de documentos contenidos en estructuras hash. Se prepara la clase, con métodos para que puedan utilizar datos diferentes a los ejemplos construidos y métodos de impresión de los datos importados para comprobaciones.

El segundo paquete, "usj.svit.topicmodeling.ui.tm", contiene las clases que se utilizarán en el proceso de conversión de los ficheros para su adaptación y que pueda procesarse mediante Topic Modeling. Se detalla a continuación.

Datos para proceso de Topic Modeling. Diseño de clases.

El diseño de la clase anterior está muy ligado al diseño del objeto que requiere el proceso de Topic Modeling en MALLET. Este objeto es, como se ha comentado en la sección 5.2.2, la instancia y la lista de ellas, pero para llegar hasta ella debe sufrir una transformación.

Una instancia en MALLET, según McCallum (39), es un objeto que contiene cuatro campos genéricos que son:

- *Data* o *datos*: contiene los datos representados por la instancia.
- *Target* o *destino*: es a menudo una etiqueta asociada con la instancia y que en aspectos de clasificación es utilizada.
- Name o *nombre*: es un nombre de identificación corto para la instancia. Ejemplo: un nombre de archivo, referencia de los datos, etc.
- *Source* o *fuente*: es la información de origen de los datos. Ejemplo: el path completo del fichero, referencia del hash, etc.

Cada campo no tiene un tipo predefinido y puede cambiar de tipo a medida que se procesa la instancia. Por ejemplo, el campo de datos puede comenzar siendo una cadena que representa un nombre de archivo y luego ser procesada por una "*pipe*" en una secuencia de caracteres que representa el contenido del archivo y, finalmente, en un vector de características

que contiene índices en un alfabeto que contiene palabras encontradas en el archivo. Depende de cada "*pipe*" qué campos en la instancia modifica; siendo el caso más común la modificación del campo de datos. Para el proyecto, el proceso crea una instancia con los datos sin procesar y se pasa a través de una "*pipe*", utilizándose los cuatro campos antes comentados.

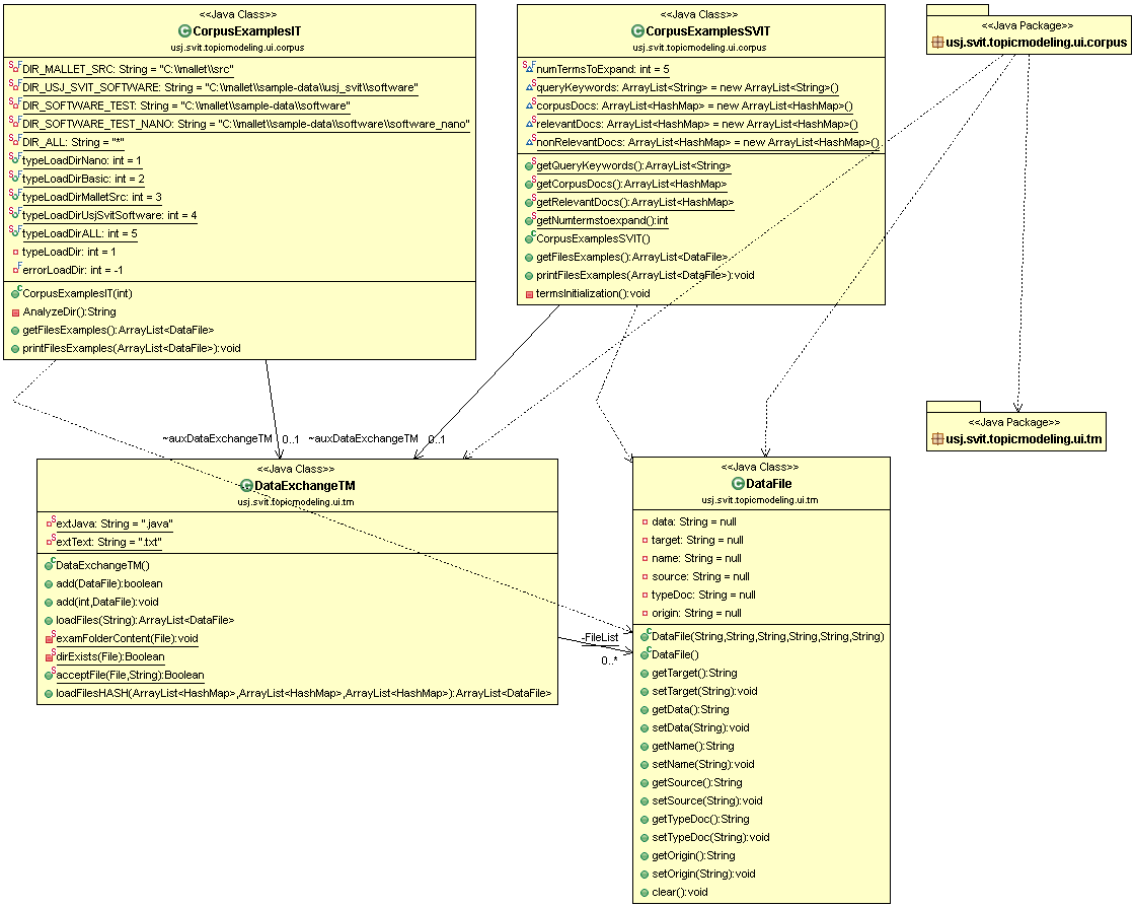


Figura 10: diagrama de clases UML para la entrada de datos y su gestión.

En el paquete "usj.svit.topicmodeling.ui.tm" de la figura anterior, se diseñan dos clases para el proceso de conversión de los datos en bruto a un objeto que luego pueda ser manejado por la clase que realice Topic Modeling. La primera clase diseñada es "DataFile". Ella proveerá objetos que contendrán los datos de cada fichero Java, texto, o documento hash conforme a los cuatro campos descritos en el párrafo anterior. La segunda clase "DataExchangeTM", es la clase que realiza la lectura de los ficheros o documentos hash y los convierte en una lista de objetos "DataFile". Este proceso se realiza con los métodos "loadFiles" para los ficheros y "loadFilesHASH" para los documentos hash. Previamente, en el caso de ficheros, se realizan comprobaciones de existencia y tipo de extensión del fichero. Los métodos se declaran públicos para usos posteriores con corpus diferentes a los ejemplos dados.

5.4.2. Sprint 5.

En este sprint del objetivo, se diseña la arquitectura que tiene la aplicación. En la reunión con el Product Owner se especifica que la aplicación forme parte de la serie de plug-in del IDE Eclipse. Se deja libertad para elegir la mejor forma de procesamiento y visualización de los datos de entrada y salida. Ver objetivo OP-02 para más detalle.

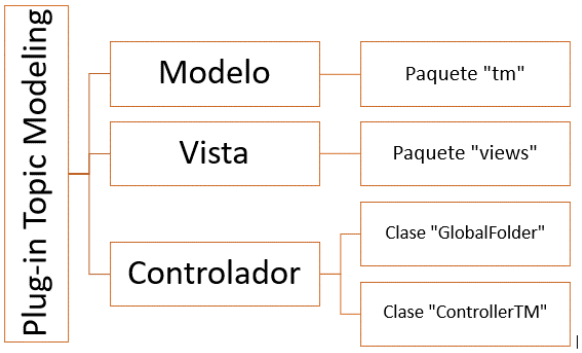


Figura 11: Esquema arquitectura MVC para plug-in

Así, gracias a optar por un plug-in, se plantea una arquitectura de tipo *Modelo Vista Controlador (MVC)*, no solamente por el hecho de tener claramente diferenciado las tres partes, sino como medida de avance en el desarrollo de la aplicación, pues al utilizar Scrum debe permitirse modificaciones y ajustes en el producto que se muestran en las revisiones del sprint. La Figura 11 muestra el esquema con los paquetes y clases asociados. Así de las tres piezas que conforman la arquitectura, ya hay una diseñada, mostrada en el sprint anterior: el modelo de datos y su gestión. Falta el diseño de la parta controladora y de visualización.

Unidad de visualización.

La Figura 12 muestra el diagrama de clases diseñado para la zona de visualización, contenida en el paquete "usj.svit.topicmodeling.ui.views". Se compone inicialmente de 5 clases:

- La clase "GlobalFolder" es quien contiene la lógica de control, junto con la clase "ControllerTM" del paquete "usj.svit.topicmodeling.ui.handlers", y parte de la gestión de la visualización.
    - Se crea el plug-in como una vista que contiene una carpeta contenedora con diferentes solapas. Cada solapa será una vista que contiene datos diferentes. Así inicialmente se plantean vistas de: datos de entrada, número de topics y su información y número de instancias y su información. Se añade una vista más que corresponde a la probabilidad de cada topic en cada instancia y una vista que muestra los créditos de la aplicación.
    - Contiene métodos de inicialización del proceso de arranque de la instancia con los datos por defecto y los "listeners" de cada "widget" del cual interesa capturar sus datos.
- La clase "About" contiene los créditos de la aplicación.
- La clase "Setup" es la clase que contiene los elementos necesarios para la interfaz entre persona-aplicación. Contiene métodos para fijar y capturar la diferente

información de todos los campos presentados. El aspecto estético será el de un formulario habitual.

- La clase "Topics" es la clase que visualiza, tras realizar el modelado, uno de los resultados principales: topic, tokens que lo forman y su probabilidad asociada, junto con la lista de instancias y la probabilidad de ese topic en cada instancia.
- La clase "TopicsDocs" es la encargada de visualizar, tras el modelado, la lista de instancias y la probabilidad de cada topic en ella.

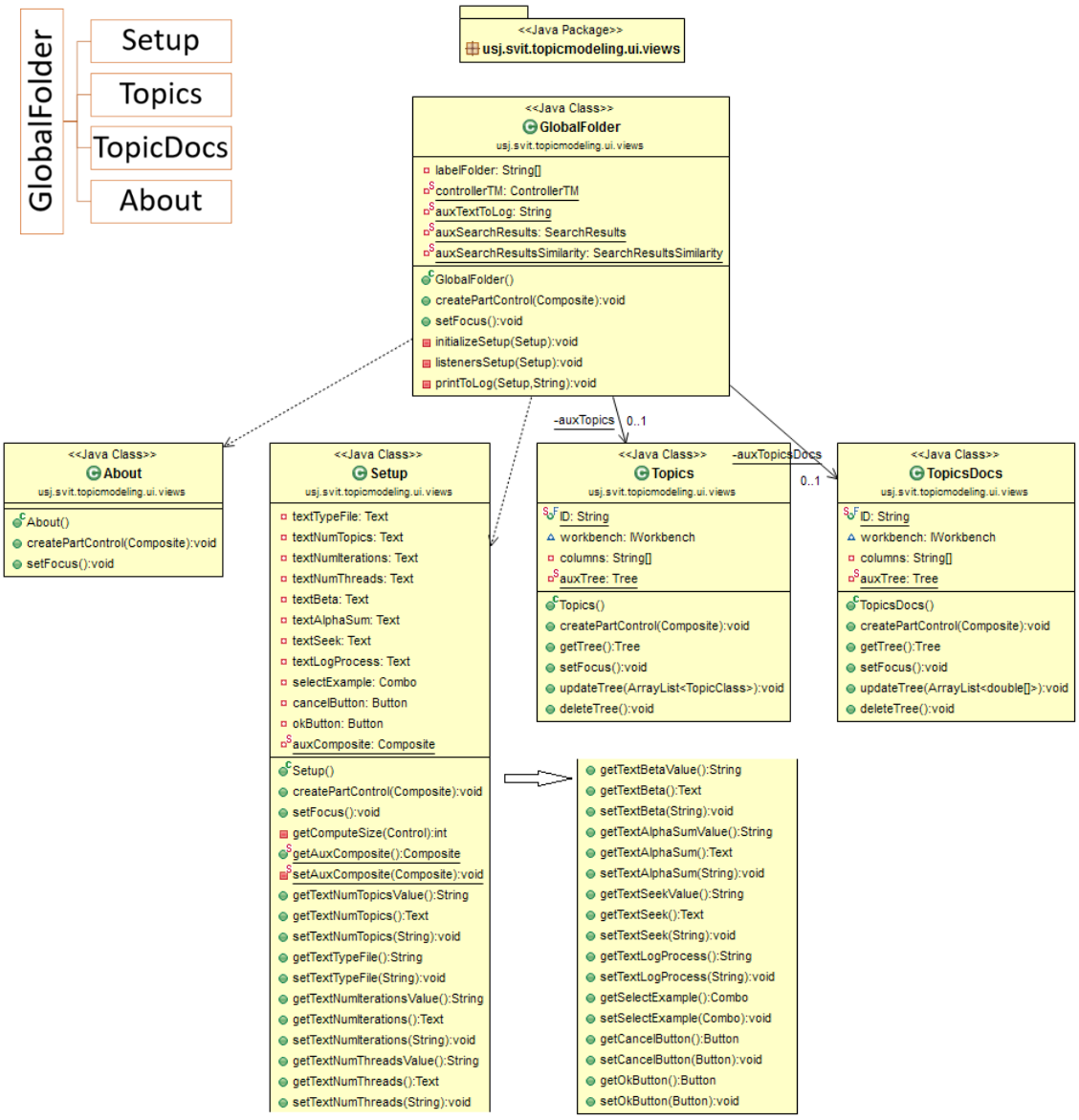


Figura 12: diagrama de clases UML para la parte de visualización del plug-in.

La Figura 13 presenta la ilustración de la interfaz del plug-in una vez planteadas las clases antes comentadas. Se observa en la parte inferior las solapas correspondientes a las clases y en la parte superior la pestaña de la carpeta contenedora.

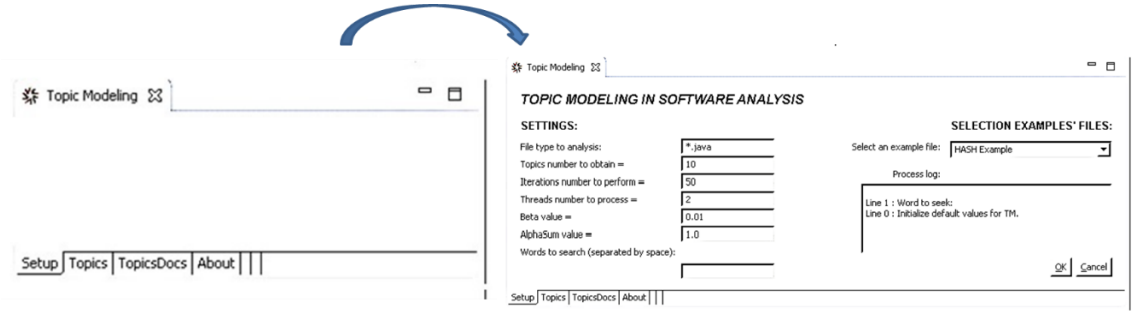


Figura 13: wireframe del plug-in. Inicial (carpeta y pestañas) y con disposición de datos.

Unidad de control.

La Figura 14 presenta el diagrama de clases diseñado para la zona de control. Se detalla la división en dos partes: la primera está contenida en el paquete "usj.svit.topicmodeling.ui.views" con la clase "GlobalFolder" y la segunda la contiene el paquete "usj.svit.topicmodeling.ui.handlers" con la clase "ControllerTM". La primera clase, además de contener los "listerners" vistos antes, utiliza un objeto de tipo "ControllerTM" para la gestión de peticiones y resultados desde-hacia el modelo de datos.

Se ha creado un sistema propio de log, que muestra la traza de movimientos y ejecución de las acciones sobre todas las vistas del plug-in. Aunque MALLET tiene uno propio, se prefiere crear uno interno por simplificación de uso y estudio.

En un primer instante, cuando se produce la carga del plug-in, se requiere la captura de los datos por defecto, para que después el cliente pueda disponer de ellos y modificarlos según conveniencia. Cada modificación de datos, aceptada por una pulsación de "enter", es comprobada en el mismo listener y si es correcta, transferida al control para la actualización.

Una vez que la clase "GlobalFolder" crea la carpeta y las solapas para las vistas, fija la solapa de la vista (clase) "Setup" y gestiona la inicialización de los valores en los diferentes campos de la misma. Para ello emplea ya métodos de la clase "ControllerTM".

Hasta este momento, todas las vistas son generadas por las clases comentadas en la subsección anterior. Solamente las vistas "Setup" y "About" tienen datos, el resto están vacías a falta de los datos que se obtienen del proceso de Topic Modeling. A este proceso se le denomina en el texto también "*modelado*" o de generación de Topic Modeling. Es puesto en marcha mediante la pulsación del botón "OK" en la vista "Setup". Todos los datos allí mostrados han sido enviados a "ControllerTM" previamente, con lo que sólo queda llamar al método "makingTM", que se encargará de todo el proceso.

5.5. OS-05: Analizar la estructura de datos que se reciben en la técnica para posteriormente aplicar Topic Modeling, muestreo y ver relaciones entre términos.

En la Tabla 11 se enumeran los detalles de trabajo realizados en este objetivo, asociados a los sprints 2 y 4. Este objetivo es complementario al objetivo anterior OP-01.

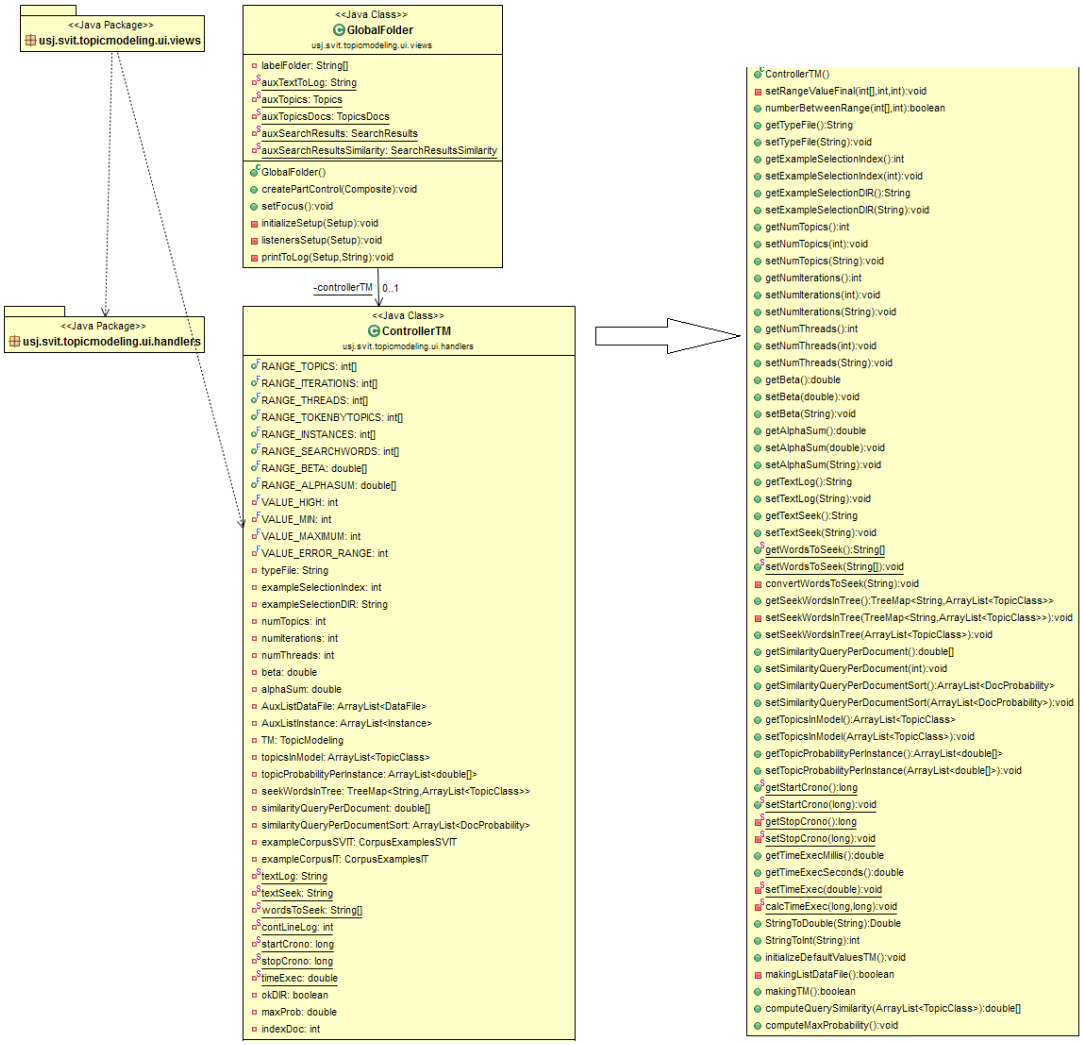


Figura 14: diagrama de clases UML para la parte de control del plug-in.

Tabla 11: OS-05. Sprint y RyR asociados.

OBJETIVO		SPRINT	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES					
ID	DESCRIPCIÓN	ID	ID	DESCRIPCIÓN	TIPO			
OS-05	Analizar la estructura de datos que se reciben en la técnica para posteriormente aplicar Topic Modeling, muestreo y ver relaciones entre términos.							
		2						
			5,00	Obtener y analizar datos de entrada tras análisis de requerimientos	Diseño			
			5,01	Elegir estructuras para los datos de entrada y probar	Diseño			
		4	5,02	Montar un corpus de ejemplo para datos de entrada	Diseño			
			5,03	Mejorar el corpus de ejemplo con más datos de entrada	Diseño			
			5,04	Obtener de Github datos de ejemplo para ampliar el corpus	Diseño			

5.5.1. Sprint 2.

Este sprint está realizado en el sprint 2 del anterior objetivo OP-01. Al definir el Product Owner los datos de entrada, en los dos formatos, se obtienen estos de manera sencilla. De una parte, se eligen una serie de ficheros de programas Java y por otra parte se crean unos simples métodos para dotar de datos a las estructuras hash. Se han comentado en el objetivo anterior los métodos y clases creadas para los ejemplos. Las pruebas realizadas consistieron en la

comprobación de lecturas y transformaciones correctas. Validadas con los métodos de impresión que figuran en las clases descritas.

5.5.2. Sprint 4.

En este sprint se trabaja con los dos tipos de datos antes comentados: ficheros tipo Java y documentos tipo hash que servirán de ejemplos para la aplicación. Se trata de aumentar la cantidad de datos, para someter la aplicación a pruebas de carga.

Documentos tipo hash.

En la Figura 10 se detalla la clase "CorpusExamplesSVIT" que contiene la estructura de tipo HashMap para almacenar los documentos y su texto. Comentado en el sprint 2 del objetivo anterior, esta clase sirve la estructura hash en forma de un "ArrayList<HashMap>, que aunque es una estructura en bruto, luego es parametrizada en el binomio <K,V> con los tipos que correspondan. Se definen tres tipos de estructuras de ejemplo: documentos de corpus, documentos relevantes del corpus y los no relevantes del corpus.

Además, se añaden estructuras que alojan las palabras de la consulta, número de términos para expandir la consulta y el objeto de tipo "DataExchangeTM", que será quien realice la conversión de la estructura hash al tipo "DataFile" estándar que se utiliza como entrada en el proceso de Topic Modeling.

Se cargan con datos todas las estructuras referidas al corpus: relevantes, no relevantes y corpus, con el método privado "termInitialization" que recibe una única llamada en el constructor de la clase. Éste produce un llenado de 6 documentos, de tipo `HashMap<String,Integer>`, cada uno conteniendo diferentes palabras, como clave y un número que identifica el número de repeticiones de cada palabra como valor.

El método "getFilesExamples" se usa para devolver la lista de ejemplos en una estructura de tipo `ArrayList<DataFile>`. Él realiza una llamada al método "loadFilesHASH" del objeto de tipo "DataExchangeTM", que es quién realiza el proceso de conversión de un tipo de estructura a otra. En el proceso hay dos inconvenientes que no pueden salvarse con la estructura de datos creada:

1. Por una parte, en el proceso de carga de datos o conversión, de los 3 grupos de HashMap (documentos corpus, relevantes y no relevantes) que acepta de parámetros, sólo se procesa el llamado "corpusDocs" o documento corpus. Los otros no pueden procesarse en sí pues sólo hay una lista de "DataFile" que luego se convierte en instancias y la estructura "DataFile" no tiene atributos para asociar ahora mismo el contenido relevante o no.

2. Debido al uso de una estructura `HashMap<String,Integer>`, ciertos campos de la clase "DataFile" no pueden ser rellenados y se dejan sin contenido. Concretamente la conversión realizada es:
    a. Campo "data": se introduce la palabra asociada a la clave tantas veces como el número de veces que aparece en el campo valor del hash.
    b. Campo "name": se califica este campo con el texto "HashCode document:" junto con el código de puntero que tiene el documento. Se opta por esta forma de identificar cada documento hash para uso posterior en las búsquedas o relaciones que se emplean.
    c. Campo "source": se califica este campo con el texto "USJ-SVIT".
    d. Los campos "target", "typedoc" y "origin" quedan vacíos.

Documentos en ficheros tipo Java.

En la Figura 10 se detalla la clase "CorpusExamplesIT" que contiene los atributos y métodos para la creación de diferentes corpus de código Java. La funcionalidad de la clase es la posibilidad de utilizar diferentes corpus para el modelado:

- Se ha creado un corpus dividido en varios directorios. Se seleccionan para cada directorio diferentes archivos JAVA y de otro tipo. Los métodos discriminarán los archivos en el proceso de conversión a "DataFile". En esta clase, se relaciona que cada directorio base es equivalente a un bloque de corpus diferente.
- Se puede seleccionar el tipo de directorio a utilizar.
- El directorio más sencillo es `DIR_SOFTWARE_TEST_NANO` que sólo contiene 2 ficheros de Java mezclados con otros ficheros en dos directorios. El siguiente directorio en complejidad es `DIR_SOFTWARE_TEST` que sube a 3 ficheros. Se realiza esta organización para las pruebas iniciales de los diferentes métodos que contiene el modelo de datos.
- El directorio `DIR_MALLET_SRC` pertenece al paquete MALLET y sólo contiene ficheros JAVA (730 Uds.) y en diferentes directorios. Se incluye porque ofrece una colección de unidades de ficheros no muy grande de tamaño y de posibles tokens que a priori se pueden conocer o es fácil relacionar de forma intuitiva. Es decir, como si fuera un entrenamiento.
- El directorio `DIR_USJ_SVIT_SOFTWARE` son una agrupación de programas JAVA que contiene más de 1.900 ficheros de tipo JAVA junto con otros tipos de ficheros en múltiples directorios. Están obtenidos en su mayoría de GitHub™ y versan sobre proyectos asociados a trenes, estando también proyectos generalistas como "JEdit" o "JTwitter". Este corpus se crea por un número mayor de ficheros a examinar y por variedad de proyectos.

- Se crea la opción `ALL` que da la posibilidad de utilizar todos los directorios en el proceso de modelado. Esta opción utilizará entonces más de 2.600 ficheros de tipo Java, con lo que la variedad de tokens para generar los topics es mucho mejor.
- La selección de directorios se realiza con un widget de tipo "combo". Utilizar este tipo de widget permite el poder indicar otro directorio de búsqueda diferente a los especificados como opción.

El método "getFilesExamples" se usa para devolver la lista de ejemplos en una estructura de tipo `ArrayList<DataFile>`. Él realiza una llamada al método "loadFiles" del objeto de tipo "DataExchangeTM". Éste realiza comprobaciones sobre el fichero (que es del tipo Java) y convierte un tipo de estructura a otra manteniendo la siguiente relación:

a. Campo "data": se introduce cada línea del fichero Java, respetando su composición de saltos de línea, acentuaciones, tabulación, etc.
b. Campo "name": se califica este campo con el nombre del fichero y extensión.
c. Campo "source": se califica este campo con el path o ruta de acceso al fichero.
d. Los campos "target", "typedoc" y "origin" quedan vacíos.

Una vez que, tanto por vía hash como fichero Java, se tiene la lista de "DataFile" completa, se está en el proceso "text extraction" que indica la Figura 8 y por tanto se avanza hacia el siguiente paso del flujo de extracción de información.

5.6. OP-02: Implementar una aplicación en Java que utilice la técnica Topic Modeling para buscar términos relacionados.

En la Tabla 12 se enumeran los detalles de trabajo realizados en este objetivo, asociados a los sprints 2, 4, 5 y 6. Esta sección contempla el segundo objetivo principal (OP-02) del proyecto.

5.6.1. Sprint 2.

En la reunión de revisión del sprint 1 y planificación del sprint 2, se detallan funciones, elementos de diseño y entrada para la aplicación. En este sprint se estudia como puede ser el primer requerimiento de la aplicación (ID-2,00): la aplicación buscará términos relacionados a partir de una consulta de uno o varios términos utilizando Topic Modeling. Se decide que la forma de trabajo será la creación de un producto mínimo viable, que se irá revisando en los diferentes sprints que restan.

En este sprint se desarrollan de forma global cuatro tareas: la primera, puesto que de OP-01 en este mismo sprint, se tienen ya definidas clases suficientes para integrar los ejemplos, será asociar éstas a paquetes o librerías, dando forma a la organización deseada y que se refleja en la Figura 10. La segunda, crear un sencillo ejemplo que pueda manejarlas y depurarlas. La tercera es estudiar e investigar cómo realizar el proceso de Topic Modeling con

MALLET y elegir el más adecuado. Finalmente, la cuarta, es estudiar la literatura asociada a la construcción de plug-in en el IDE Eclipse. En este sprint se muestran la segunda y tercera.

Tabla 12: OP-02. Sprint y RyR asociados.

OBJETIVO ID	OBJETIVO DESCRIPCIÓN	SPRINT ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES DESCRIPCIÓN	TIPO
OP-02	Implementar una aplicación en Java que utilice la técnica Topic Modeling para buscar términos relacionados				
		2			
			2,00	La aplicación buscará los términos relacionados a partir de una query compuesta de términos y devolverá los topics relacionados con ella tras aplicar la técnica de TM a los documentos del corpus.	Función
			2,01	La aplicación en Java tiene que tener unos métodos que permitan su utilización desde otros proyectos Eclipse	Función
			2,06	La aplicación requiere ver si los términos están en el alfabeto y en caso positivo realizar la relación (cálculo de probabilidades)	Función
			2,10	Montar paquetes con las clases definidas	Diseño
			2,11	Montar un programa "main" con los objetos definidos	Diseño
			2,12	Pruebas sobre ficheros JAVA y clase tm (Topic Modeling)	Pruebas
			2,13	Estudiar diferentes métodos de MALLET para realizar TM y elegir el más adecuado	Pruebas
			2,20	Eclipse: búsqueda de literatura sobre plug-in	Organización
			2,21	Eclipse: leer y aplicar ejemplos de plug-in	Organización
			2,80	Cada documento HashMap estará formado por términos y sus frecuencias y/o con la información que requiera TM	Entrada
			2,81	Los documentos (ficheros) de entrada serán HashMap o tipo .JAVA y estarán en idioma inglés	Entrada
		4			
			2,14	Validar las pruebas con programa TM "main"	Pruebas
			2,22	Crear un plug-in de ejemplo	Pruebas
			2,23	Analizar plug-in de ejemplos vistos (Reformulation, Traductor,…)	Diseño
			2,24	Crear un sketch del plug-in	Diseño
		5			
			2,31	Ampliar la zona de vistas. Pruebas con diferentes widgtes y layout. Ampliación para resultados búsquedas.	Desarrollo
			2,32	Ampliación zona de controlador. Ampliacion para resultados búsquedas	Desarrollo
			2,90	Como usuario se requiere una probabilidad de cada consulta sobre los documentos para ver el documento con el contenido más similar	Función
			2,91	Como usuario se requiere un botón para la ejecución del proceso de cálculo de TM	Función
			2,92	Como usuario se requiere un botón para la ejecución del cálculo de búsqueda de términos relacionados	Función
		6			
			2,33	Integrar las clases de TM realizadas como parte de modelo en la arquitectura	Desarrollo
			2,34	Ajustes del modelo al controlador	Desarrollo
			2,93	Como usuario se necesita una vista dónde se muestre el resultado de los documentos que tienen más probabilidad de contener la secuencia de búsqueda.	Función

Ejemplo de uso de paquetes de Corpus.

Se crea un sencillo programa Java que hace uso de las clases de la Figura 10 para comprobar su uso y generación del ArrayList <DataFile> correcto para cada uno de los ejemplos. Se utiliza el método "printFilesExamples" como herramienta de ayuda para la comprobación. La Figura 15 muestra ejemplos del resultado de la importación de ambos tipos.

Métodos empleados para realizar Topic Modeling en MALLET.

MALLET tiene algoritmos implementados específicamente para realizar Topic Modeling, Clasificación, Clustering o Modelos de Predicción de Secuencias entre otros. Para realizar Topic Modeling, MALLET tiene una serie bastante completa que comprenden algoritmos como LDA, Paralell LDA, DMR LDA, Hiercharchical LDA, Labeled LDA, etc. Que permiten realizarlo adaptado a los requisitos que se busquen.

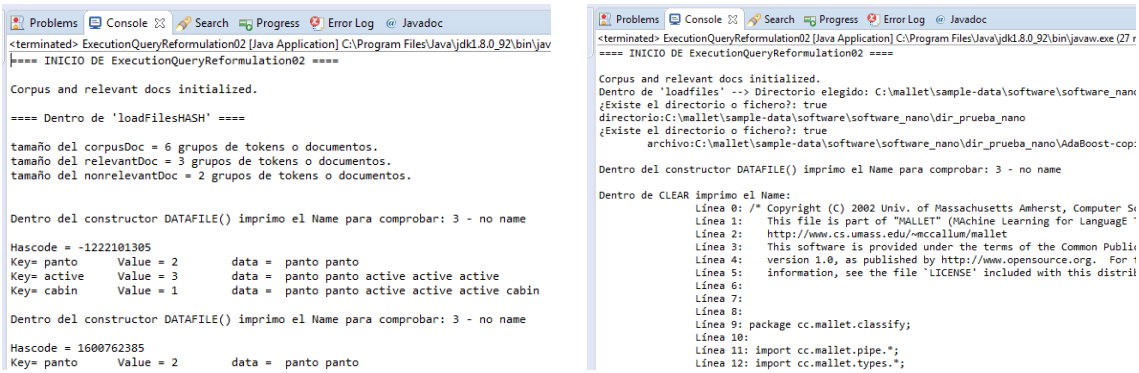


Figura 15: captura de importación de datos en hash y fichero Java en Inicio de ExecutionQueryReformulation02.

Para este objetivo, en el proceso de preparación del sprint, se acuerda con Product Owner, centrar el uso en el algoritmo LDA, sin desviaciones a otras tipologías, debido a que se trata de usar Topic Modeling para búsquedas, sin entrar en el estudio de sus variantes. Para ello se examinan en profundidad los diferentes métodos que contemplan LDA de una manera básica. Se presentan seguidamente tres métodos seleccionados, comentando el elegido.

1 Método SimpleLDA.

McCallum (39) comenta que es una implementación simple de LDA utilizando el muestreo de Gibbs. Referencia que el código es más lento que la implementación normal de Mallet LDA (segundo método comentado), pero proporciona un mejor punto de partida para comprender cómo funciona el muestreo y para construir nuevos modelos de temas. Al ser tan sencillo la implementación parte de la base de un alfabeto ya generado y depurado para su proceso.

2 Método LDA (obsoleto).

McCallum (39) creó inicialmente, junto a su grupo, este método como implementación del algoritmo LDA desarrollado por Bley et al. (45) y utilizando muestreo de Gibbs como técnica para el proceso de cálculo. Se califica de obsoleta por el mismo autor, debido a las limitaciones que tiene como: añadir más documentos de forma incremental, manejar la secuencia de características "FeatureSequence" de forma indirecta, y necesitar un soporte para el crecimiento del vocabulario. Como corolario McCallum comenta usar el método "ParallelTopicModel".

3 Método ParallelTopicModel.

McCallum (39) comenta que es un desarrollo de David Mimno y él, con la característica que es una implementación de LDA básica pero con procesamiento en paralelo (con threads) y basada en los artículos de Newman et al. (46) y con un esquema de muestreo de tipo "SparseLDA" desarrollado por Yao et al. (47). La gran diferencia es que mejora el método anterior y además permite la configuración de más parámetros que la anterior. En conclusión,

se utiliza el método "ParallelTopicModel" por ser un método basado en LDA, pero con capacidad de procesamiento en paralelo y mayor configuración de variables que los anteriores.

5.6.2. Sprint 4.

En este sprint se contemplan varias tareas: crear un sketch o bosquejo del plug-in a desarrollar, crear un plug-in básico de ejemplo, tras la lectura de la bibliografía y validar las pruebas con el programa de importación de ficheros.

Validación de las pruebas de importación.

En Figura 15 se muestran los resultados del proceso de importación de los dos tipos de corpus diseñados. En el tipo hash, vista izquierda, se aprecian la repetición de las palabras, generando un dato mayor que el recibido, pero con la intención que dichas palabras tengan mayor peso en el proceso de modelado que las que tienen menos repeticiones. Se aprecia en la parte izquierda el binomio <K,V> y en la derecha la multiplicación de las palabras. En el tipo fichero Java, vista derecha, el proceso es más simple, al tratarse de una copia directa de las líneas del fichero en el objeto "DataFile" atributo "data".

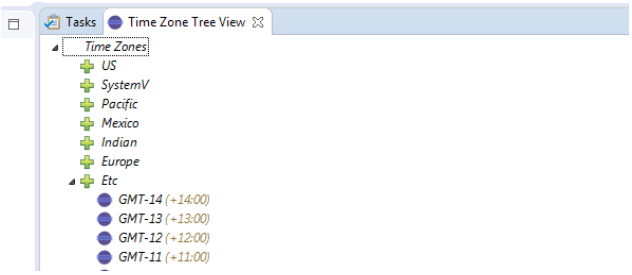


Figura 16: vista plug-in de ejemplo desarrollado.

Plug-in de ejemplo.

Tras la lectura de Blewitt (48), se realiza un plug-in de ejemplo, según los comentarios del libro. La Figura 16 presenta dicho ejemplo en una vista en forma arborescente de control de zonas horarias. Este plug-in sirve para determinar el uso de los árboles como estructura para mostrar los resultados del proceso de modelado, al poder visualizar agrupaciones de palabras asociadas a un nodo del árbol y cada nodo representar un topic. También se asocia que cada nodo sea una instancia y sus ramas o hijos la probabilidad de cada topic. Esta disposición permite por último poder crear estructuras más complejas, como asociar las instancias a topics con probabilidades.

Sketch del plug-in.

Salgado (49) comenta que un sketch es un boceto primario que se realiza para un proyecto digital y son unos trazos sobre una hoja de papel. Tras el ejemplo realizado en Figura 16, se realiza un boceto con los requisitos, requerimientos y funcionalidades, que se tienen hasta ahora, que se representa en la Figura 17. Del boceto se procede a dar forma en un

"wireframe" como muestra Figura 13. Desde ellas se pasa directamente al prototipo omitiendo el "mockup". Puesto que se está bajo Scrum, es más conveniente hacerlo de esta forma, para ofrecer al Product Owner un producto que pueda probar en cada revisión, viendo las mejoras y cambios para implementar en el siguiente sprint.

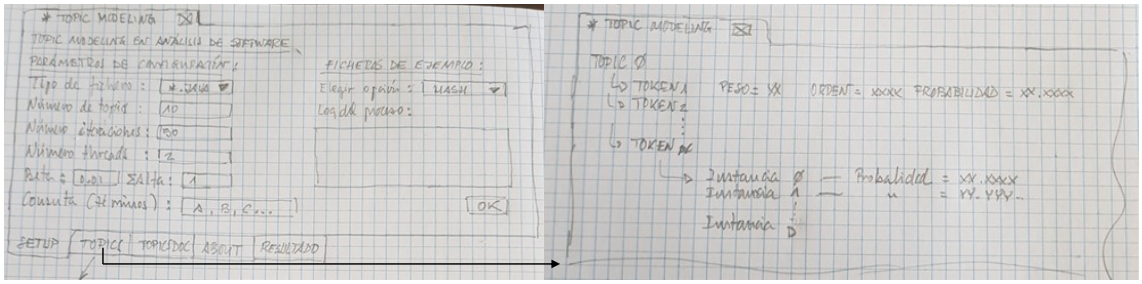


Figura 17: boceto o sketch del plug-in a realizar.

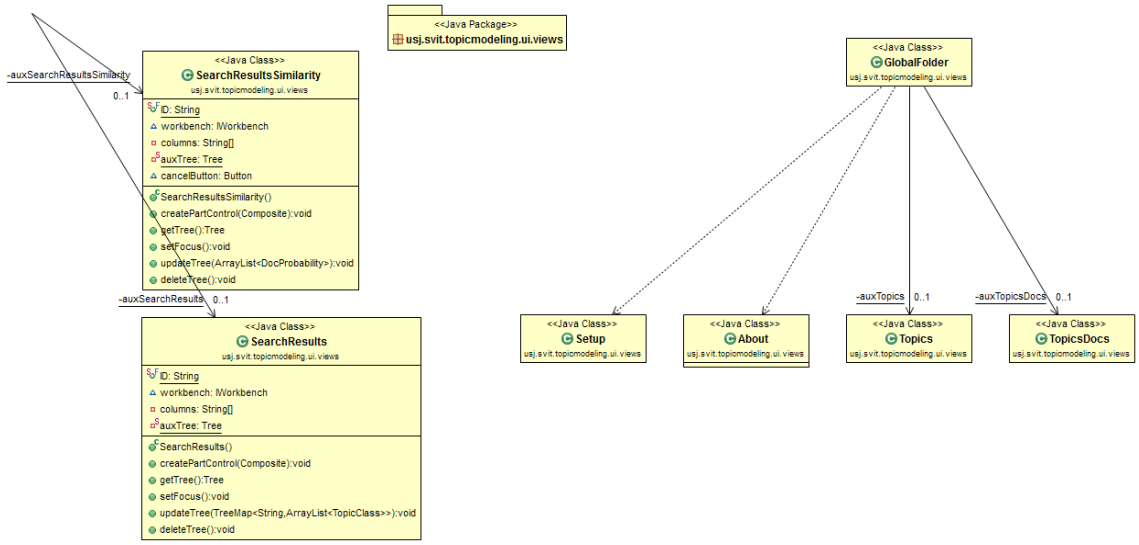


Figura 18: diagrama de clases UML para la parte de visualización del plug-in. Ampliación.

5.6.3. Sprint 5.

Este sprint, es un complemento del objetivo OP-01 sprint 5, al ocuparse de la implementación y ampliación de requisitos y funcionalidades. Concretamente se contemplan varias tareas: La primera es la ampliación de la zona de vistas y controlador en función del requerimiento de búsquedas relacionadas. La segunda es la integración del proceso de Topic Modeling. Es en este proceso dónde el flujo estipulado en la Figura 8 se termina y se da lugar al proceso de modelado mediante el método "ParallelTopicModel".

Ampliación de paquetes y clases asociadas a vista y controlador.

Como indica la Tabla 12, en este sprint se incorporan nuevas funcionalidades que afectan a los paquetes "usj.svit.topicmodeling.ui.views" y "usj.svit.topicmodeling.ui.handlers".

La Figura 18 presenta el paquete de las vistas incorporando dos nuevas clases. Son las clases "SearchResults" y "SearchResultSimilarity" que se encargan de mostrar en vista arborescentes

los resultados de los dos tipos de búsquedas. Se omite la descripción de las clases que ya estaban referenciadas en Figura 12.

La Figura 19 presenta el paquete de control o "handlers" dónde se añade una clase más, "DocProbability" para soportar los cálculos que se utilizarán en la búsqueda de tipo "similarity", junto con nuevos métodos en la clase "ControllerTM". Se omite la descripción de las clases que ya estaban referenciadas en Figura 14.

Integración del proceso de Topic Modeling.

En esta parte del sprint se procede a la finalización del proceso de tratamiento de los textos y puesta en marcha del modelado o proceso de Topic Modeling. Se tiene de los objetivos y sprints anteriores, una estructura de datos con los textos y su información asociada en bruto.

Figura 19: diagrama de clases UML para la parte de control del plug-in. Ampliación.

Del estudio del método "ParallelTopicModel" se obtienen los parámetros con los cuales se llama al constructor de la clase. En este caso se elige llamar al constructor con los parámetros siguientes:

- Número de topics: es un entero, que especifica el número de topics a generar. Por defecto el plug-in tomará un valor de 10 topics.
- Sumatorio de alfa[15] o `alphaSum`: es un real (double) que encarna, la suma de valores alfa. Por defecto, el plug-in colocará valor igual 1 o lo que es lo mismo valor de 0,1 sobre cada topic, si se toma un valor de topic igual a 10.

15 Según Bathia (9), alfa, es la probabilidad que cada documento contenga una mezcla de la mayoría de los temas, y no un solo tema en particular. Un valor alfa bajo pone menos de tales restricciones en los documentos, y esto significa

- Beta[16]: es real (double), por defecto, el plug-in colocará valor igual a 0.01.

Además, la clase tiene otros parámetros por defecto, como el número de palabras por topics (7), el número de iteraciones (1.000), periodo de burn-in (200), etc. Que pueden ser alterados con los métodos correspondientes. En este punto, se tienen ya los parámetros para el método, pero falta cómo especificar el conjunto de datos a procesar, es decir la entrada.

Del objetivo OS-05 se comprueba que la estructura de datos creada es correcta, como corpus leído, pero no es admitida por la clase "ParallelTopicModel" como entrada para el proceso de Topic Modeling. Esta clase necesita, como entrada, una lista de instancias para poder iniciar el proceso de modelado. Para convertir la lista de objetos "DataFile", dónde cada uno de ellos es un documento, a una lista de instancias, dónde cada instancia representará a un documento, se tienen que realizar los pasos indicados en Figura 8. Es el llamado *procesamiento del texto*.

El método "makingTM" de la clase "ControllerTM", realiza las llamadas a los diferentes métodos para la generación del proceso. La activación de este método se genera con la pulsación del botón "OK" en la vista "Setup" que corresponde con los requerimientos ID-2,91 e ID-2,92. Los diferentes métodos que son llamados desde "makingTM" en secuencia son:

1. En primer lugar, llama al método "makingListDataFile" del mismo paquete para realizar la conversión de los ficheros de corpus en un `ArrayList<DataFile>` comprobando si el resultado es correcto o no.
2. En caso de ser correcto, se fijan las listas de palabras a eliminar de los textos para el procesamiento posterior.
3. Se realiza la transformación de `ArrayList<DataFile>` a un `ArrayList<Instance>` llamando al método "transformDataFileToInstances" del paquete "usj.svit.topicmodeling.ui.tm". Ver Figura 20. Cada ítem de la lista tiene una instancia que es un objeto que contiene los cuatro campos: `data`, `target`, `name` y `source`.
4. Esa lista de instancias debe procesarse para generar una nueva lista de instancias de tipo `InstanceList`. Esta es la que procesará "ParalellTopicModel" como dato de entrada, siendo en este punto dónde se realiza la depuración del texto o el procesamiento descrito en Figura 8. Para ello se utiliza el método "makeInstanceList" del paquete "usj.svit.topicmodeling.ui.tm". Éste utiliza un objeto de tipo `Pipe`, que es

que es más probable que un el documento puede contener una mezcla de solo algunos o incluso uno de los temas. Es decir, el valor es realmente es una suma sobre la probabilidad de la mezcla de topics (temas).

16 Según Bathia (9), un valor beta alto significa que es probable que cada topic contenga una mezcla de la mayoría de las palabras, y ninguna palabra específicamente; mientras que un valor bajo significa que un topic puede contener una mezcla de sólo algunas de las palabras. Según McCallum (39) es un parámetro previo a la distribución multinomial por topic sobre las palabras.

un tipo especial que tiene MALLET dedicado al procesamiento de textos. La generación del objeto `Pipe` se realiza con el método "buildPipe" del paquete "usj.svit.topicmodeling.ui.tm". Dicho método crea una "pipe" o cadena de procesamiento, en principio vacía, para el texto que contiene la instancia.

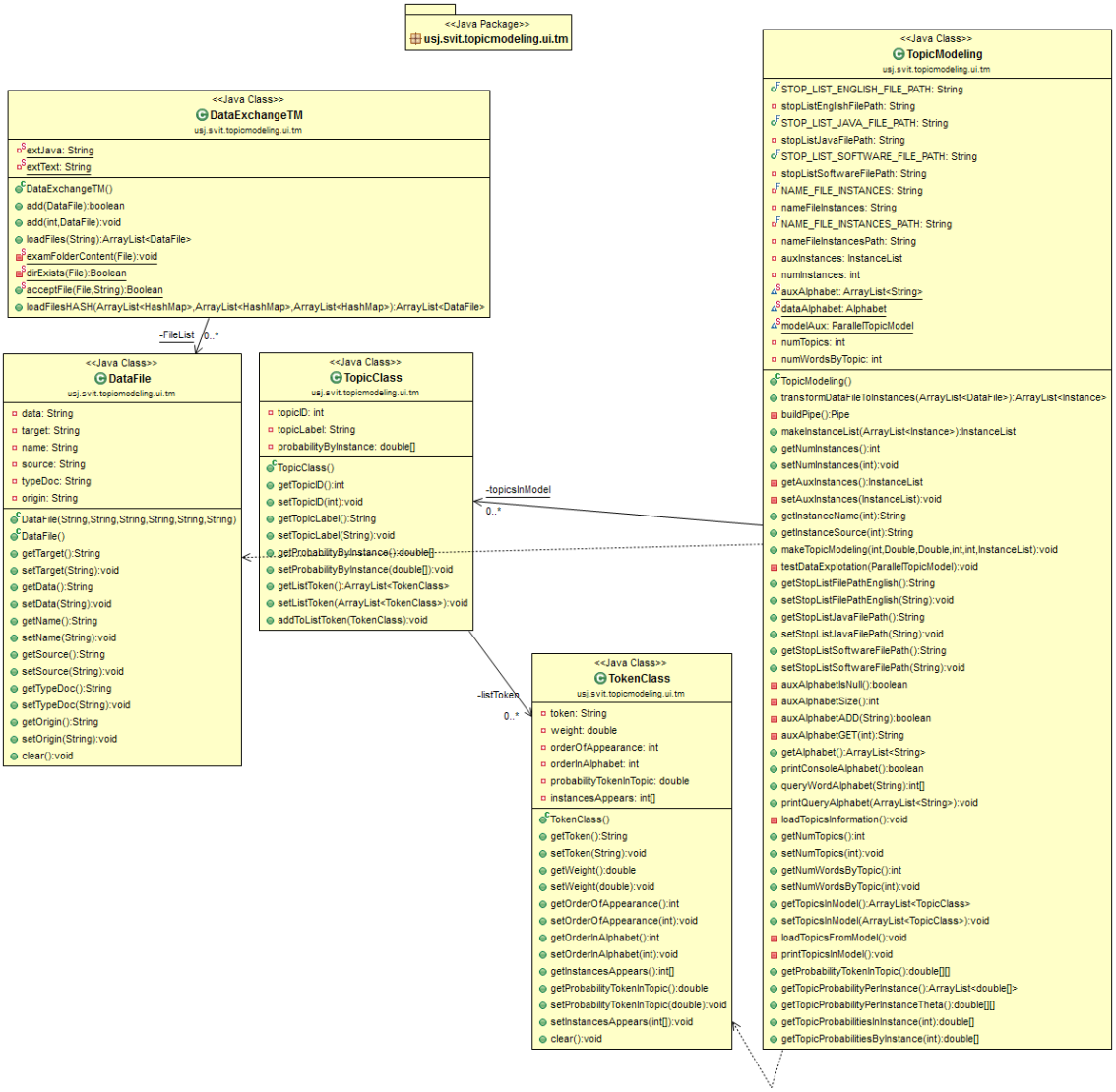


Figura 20: diagrama de clases UML para la parte del modelo de datos del plug-in.

5. Con esa `InstanceList`, número de topics, `alphaSum` y beta, ya está todo preparado para la llamada al método "makeTopicModeling" del paquete "usj.svit.topicmodeling.ui.tm" e iniciar el proceso de Topic Modeling utilizando un objeto de la clase "ParallelTopicModel", que es quién va a gestionar todo el proceso de modelado. En la llamada al constructor se pasan los parámetros de número de topics, `alphaSum` y beta. La finalización del método carga, con la llamada al método "loadTopicsInformation", los datos obtenidos en los diferentes objetos que luego serán

manipulados en las vistas. En Figura 20 se describen esos objetos, que pertenecen a las clases "TopicClass" y "TokenClass". La creación de estas clases se origina como medida para tener asociada a una clase y por tanto a objetos de ésta los resultados del proceso de Topic Modeling, que arroja una matriz de topics y palabras asociadas, junto a sus probabilidades, llamada *matriz fi (Φ)* y otra de documentos (instancias) con los topics y sus probabilidades llamada *matriz zeta (θ)*. Con estas clases dotamos a ambas matrices de una relación entre ellas y se añaden más atributos a los topics para una mejor explotación de datos posterior. La estructura que se maneja es de tipo `ArrayList<TopicClass>`. La matriz *fi* es el método "getProbabilityTokenInTopic" y la matriz *zeta* es el método "getTopicProbabilityPerInstaceTheta".

6. La salida el método "makingTM" carga los datos en los objetos para las vistas "Topics" y "TopicsDocs". Esos objetos son manipulados por el método "updateTree" de ambas vistas, llamando a los métodos "getTopicsInModel" para la vista "Topics" y "getTopicProbabilityPerInstance" para la vista "TopicsDocs" y serán utilizados todavía dentro del listener de la pulsación del botón "OK". La Figura 21 muestra una imagen de ambas. En la izquierda se muestra la vista "Topics" y a la derecha "TopicsDocs".

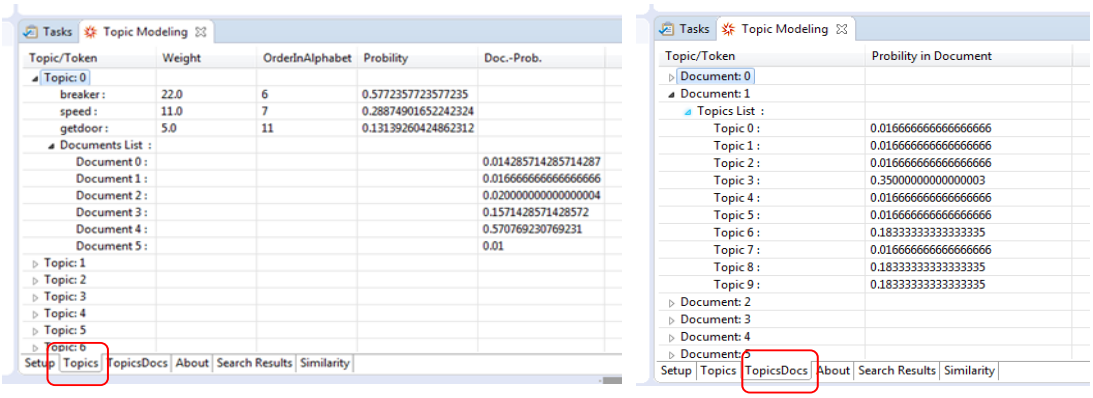


Figura 21 : solapas o pestañas "Topics" y "TopicsDocs" tras proceso de modelado.

La salida del listener de la pulsación del botón "OK", acaba con la actualización de la vista "SearchResults", mostrando los resultados del requerimiento ID-2,00, es decir, los topics, con sus términos, relacionados con la cadena de búsqueda introducida. El proceso se inicia con la llamada al método "setSeekWordsInTree" de la clase "ControllerTM" y de parámetro se le pasa la lista conteniendo los topics y toda su información, es decir, un `ArrayList<TopicClass>`. Éste recorre la lista de topics y con cada coincidencia de un token, que forma el topic, con alguna de las palabras de la cadena, añade ese topic y su información a un árbol, con estructura de tipo `TreeMap<String, ArrayList<TopicClass>>`, para posteriormente ser representado. En é, el valor clave, representa la palabra de la cadena de búsqueda que coincide con algunas de las que forman el topic. Como la palabra puede estar en varios topics, se asocia la lista de todos ellos que la

contienen. Se finaliza con la llamada al método "updateTree" y pasando como parámetro la estructura TreeMap anterior, devuelta por el método "getSeekWordsInTree" de la clase "ControllerTM". En la Figura 22 se muestra el resultado de la ejecución del método en la vista "SearchResults" en la ejecución de la prueba 1 descrita en el capítulo 7.

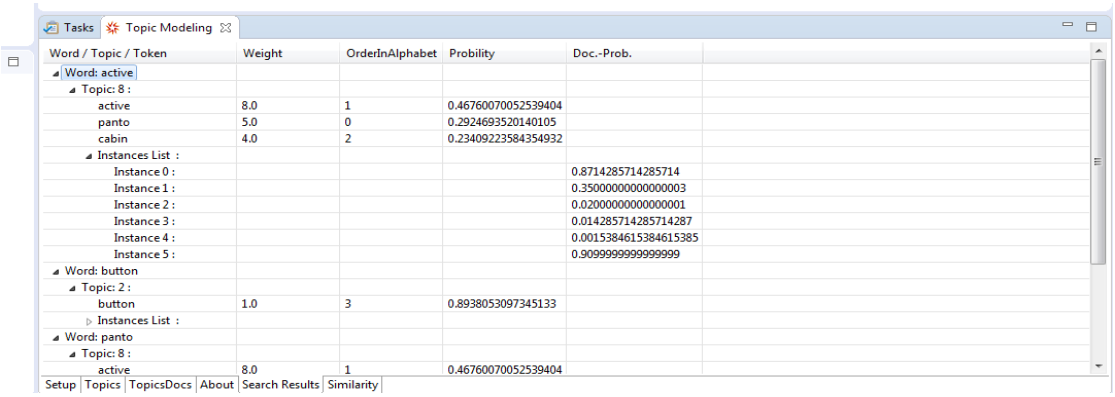


Figura 22: Vista "Topic Modeling". Solapa "SearchResults". Resultado en Prueba 1.

5.6.4. Sprint 6.

En este sprint se contemplan varias tareas: terminar la integración del modelo de datos con el resto de la arquitectura, como ayuda a esta tarea, se modifica o ajusta el controlador y una nueva función asociada al sistema de búsqueda de términos.

Integración final del modelo datos.

Según se observa en Figura 19 y Figura 20, en comparación con Figura 14, la integración final del controlador con el modelo de datos se realiza mediante dos actuaciones. Primero, se crean las clases "TopicClass" y "TokenClass" por parte de la zona de modelado, junto con la clase "DocProbability" de la zona de control. Estas clases dan soporte para la explotación de datos una vez realizado el modelado. Segundo, se crean los métodos asociados a esas clases para su manejo, y nuevos métodos y atributos, como los análisis de cadena de búsqueda (métodos asociados a la denominación "WordsToSeek"), control de tiempo del proceso (métodos asociados a la denominación "Crono"), análisis de probabilidades (métodos asociados a la denominación "TopicProbability"), manejo del alfabeto (métodos asociados a la denominación "Alphabet"), manejo y configuración de topics (métodos asociados a la denominación "Topics") etc. Que dan soporte a la gestión de los datos tanto de entrada como de salida.

Nueva función "similarity".

En la reunión de revisión del sprint 5 y preparación del 6, se añade una nueva función al plug-in, requerimiento ID-2,96: crear una vista dónde se muestre el resultado de los documentos que tienen más probabilidad de contener la secuencia de búsqueda. Observando la

Figura 18 se presenta el resultado en la vista "SearchResultsSimilarity" del paquete "usj.svit.topicmodeling.ui.views".

Si bien la vista creada en el sprint 5 ofrece un resultado sobre la búsqueda multipalabra de la secuencia de búsqueda, ver Figura 22, el resultado que presenta es por cada término o palabra y no para el conjunto total de ellos. Ahora en esta nueva funcionalidad, se requiere referenciar qué documento (instancia) es el más probable que contenga todos los términos o palabras de la búsqueda. No sólo se fija sobre un documento, sino que se da la relación de documentos, ordenados de mayor probabilidad a menor probabilidad. A este proceso se le denomina "*similarity*" o *semejanza* y se calcula según las ecuaciones siguientes:

Ecuación 1

$$P\ (w|d) = \sum_{t=1}^{T} p(w|t) * p(t|d)$$

Ecuación 2

$$Sim(Q, D_i) = P(Q|D_i) = \prod_{q_k \in Q} P(q_k|D_i)$$

La semejanza se expresa como la probabilidad condicional de una consulta sobre un documento D_i y q_k es la enésima palabra en la consulta. Se tiene como origen la propia consulta Q, los topics y su relación con los documentos, matriz zeta (θ), y las palabras o tokens en los topics, matriz fi (Φ). La Ecuación 1 es la probabilidad de esa palabra en los documentos, que se expresa en la implementación como la multiplicación de la matriz Φ por la matriz θ para cada topic, es decir, será el sumatorio de la multiplicación del valor de probabilidad de la palabra en ese topic por la probabilidad de ese topic en el documento dónde aparece. Este procedimiento de cálculo se implementa en el método "computeQuerySimilarity" de la clase "ControllerTM". Mientras que la Ecuación 2 calcula la semejanza, como el producto de probabilidad de la celda (D_i) del vector "*semejanza_por_documento"* por el valor de la probabilidad de cada palabra por documento calculada en cada documento por la anterior ecuación.

Este método se añade al final del listener del botón "OK". Tras esa llamada, se tiene un vector de documentos y probabilidades de la cadena de búsqueda en cada uno. Posterior a este proceso se usa el método "computeMaxProbability" de la clase "ControllerTM", que realiza una ordenación del vector que contiene la semejanza por documento y se le añade más información: nombre y ruta al documento, creando una lista de objetos "DocProbability". Tras la ordenación, se usa el método "updateTree" para pasar los datos a la vista "SearchResultsSimilarity" y como parámetro se pasa la lista de objetos "DocProbability" obtenida con el método "getSimilarityQueryPerDocumentSort" de la clase "ControllerTM". La Figura 23 presenta un ejemplo de la solapa "Similarity" tras la ejecución del modelado y consulta de términos en la prueba 1 del capítulo 7.

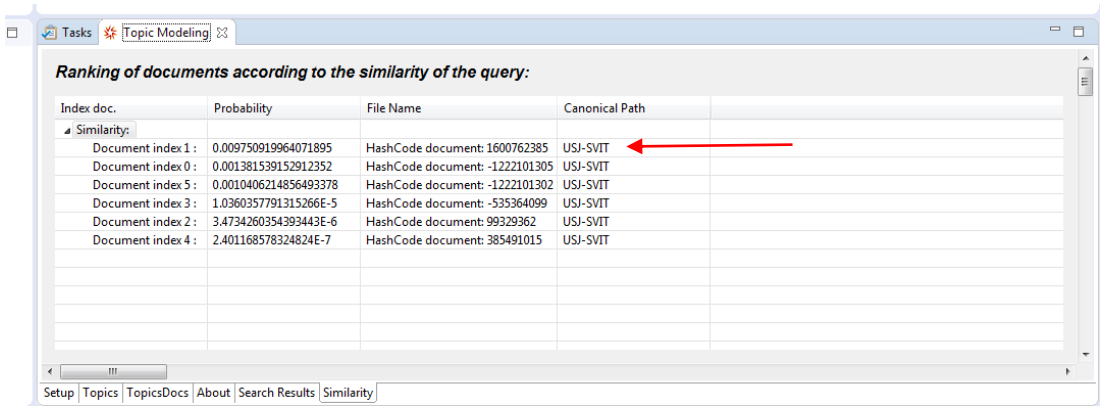


Figura 23: Vista "Topic Modeling". Solapa "Similarity". Resultado en Prueba 1.

5.7. OS-06: Establecer escenarios en los que se implementará y analizará la aplicación Java para evaluar su comportamiento.

En la Tabla 13 se enumeran los detalles de trabajo realizados en este objetivo, asociados al sprint 6.

Tabla 13: OS-06. Sprint y RyR asociados.

OBJETIVO		SPRINT	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES		
ID	DESCRIPCIÓN	ID	ID	DESCRIPCIÓN	TIPO
OS-06	Establecer escenarios en los que se implementará y analizará la aplicación Java para evaluar su comportamiento				
		6			
			6,00	Escenario 1: origen de datos vía HashMap	Entrada
			6,01	Escenario 2: oriden de datos vía fichero JAVA	Entrada

5.7.1. Sprint 6.

En la reunión de revisión del sprint 5 y planificación del sprint 6, se detallan como elementos de entrada dos escenarios posibles (ID-6,00 e ID-6,01) sobre los cuales se realiza la aplicación y pruebas. Como se ha comentado en otros objetivos o sprints vistos, el origen de los datos viene dado por dos tipologías diferentes de entrada de datos: ficheros de tipo Java y estructura de datos tipo hash, aunque en el procesamiento posterior dichas entradas acaban siendo tokens y se igualan, se vuelven a definir en este sprint para establecer la base al siguiente objetivo, OS-07, de evaluación del comportamiento.

6. ESTUDIO ECONÓMICO.

En este capítulo se presenta el estudio económico del PFG.

6.1. Presupuesto.

En Tabla 15 se muestra el desglose del presupuesto del proyecto. Éste se divide en diversos capítulos, a saber:

- *Personal*: dónde se refleja el coste de la persona que realiza el proyecto, es decir, proyectando, que actúa como Scrum Master y desarrollador o equipo de desarrollo o "developer team". Se contemplan las horas destinadas al trabajo según la "*guía docente de la asignatura 30501*" de la Universidad. La parte de propietario del producto o Product Owner la asume la directora de proyecto y al ser la cliente no se imputa coste de personal al presupuesto.

  Para el cálculo del precio hora, se ha tomado como referencia el precio o coste de hora laboral[17] ofrecido por el Instituto Nacional de Estadística de España en referencia al cuarto trimestre del 2019. Tabla 14. Este coste se obtiene trimestralmente de la "*Encuesta Trimestral de Coste Laboral*" (E.T.C.L.) consultando las diferentes empresas cuyo C.N.A.E. 62 corresponde a "*las empresas de programación, consultoría y otras actividades relacionadas con la informática*" y se suele publicar con un trimestre de retraso, de ahí el empleo del último trimestre del año 2019.

  Si bien la encuesta no refleja costes por categoría profesional, tampoco se aplica referencia a algún convenio salarial.

- *Amortización de los equipos empleados y software licenciado*: para la realización del proyecto, se ha empleado un portátil y preparado un ordenador virtual diferente que actúa en caso de avería del primero o para salvaguarda. Sólo se tiene en cuenta el primero para el coste de amortización. Para ello se ha tenido en cuenta un coste estimado según precio de mercado. Además, se aplica un periodo de depreciación de

17 Según I.N.E. en su glosario (56), define *coste laboral* como "l*os costes laborales consisten en el gasto total que soportan los empleadores al emplear a su personal, concepto que ha sido adoptado en el marco comunitario y que se ajusta globalmente a la definición internacional de la Conferencia Internacional de Estadísticas del Trabajo (Ginebra, 1966). Los costes laborales comprenden la remuneración de los asalariados (D.1) con sueldos y salarios en efectivo o en especie, las cotizaciones sociales a cargo de los empleadores, los costes de formación profesional (D.2), otros gastos (D.3) y los impuestos relacionados con el empleo considerados costes laborales (D.4); debe deducirse toda subvención recibida (D.5). Los costes correspondientes a personas empleadas por agencias de empleo temporal deben incluirse en la rama de actividad de la agencia que las emplea (NACE Rev.2, 78.20) y no en la rama de actividad de la empresa para la que trabajan realmente*". También en Eurostat (53) puede verse una explicación mejor detallada de los componentes del coste laboral.

En resumen, el coste laboral es el coste total que incluye el empleador por la utilización del factor trabajo, siendo la suma del coste salarial más otros costes derivados como son: pagos en especie, cotizaciones sociales a cargo del empleador, coste de formación profesional, otros gastos y los impuestos asociados a los costes laborales.

Existen otras referencias salariales tanto públicas como privadas (*convenios*) que no se han tenido en cuenta en el proyecto, pero que sí muestran diferentes categorías profesionales y que no son de aplicación en este proyecto.

48 meses[18] (4 años) y un porcentaje de uso para el proyecto frente a las horas dedicadas por persona. Así se logra ajustar el periodo de amortización al verdadero uso para el proyecto. El ordenador tiene en el precio de compra incluida la licencia de sistema operativo. El resto de las licencias a emplear en el proyecto serán todas de software en modo código abierto, licenciadas por sus respectivos fabricantes.

Tabla 14. Coste laboral por hora CNAE-09.62. Fuente: I.N.E. (50).

Resultados nacionales (desde el trimestre 1/2008)
Componentes del coste laboral total

Coste laboral por hora efectiva por divisiones de la CNAE-09
Unidades: Euros

	Coste laboral total por hora			
	2019T4	2019T3	2019T2	2019T1
62 Programación, consultoría y otras actividades relacionadas con la informática	28,39	28,20	25,49	23,49

Notas:
Consultado el 16-04-2020 16:59
Fuente:
Instituto Nacional de Estadística

- *Otros costes directos*: en esta partida se refleja los gastos de viaje que va a tener el proyecto, un viaje del proyectando a la sede de la Universidad para la defensa del proyecto, junto con los materiales empleados en el mismo y unos libros.
- *Resumen*: Se presenta en este apartado el resumen de los apartados anteriores junto a una partida más:
    - La partida de "*margen bruto*" formada por dos conceptos más que son: los costes indirectos y el beneficio de la venta. Como estimación del primero se ha tomado una media, a la baja, de referencia, según experiencia del proyectando, de otros proyectos industriales y la calificación de costes indirectos de ayudas a proyectos de la Unión Europea. El segundo valor, beneficio, se ha estimado en cero puesto que no hay lucro con este proyecto.

Al final de la tabla se muestra la suma del presupuesto, que en este caso asciende a *10.953,00* euros.

18 La Hacienda Pública española autoriza una amortización anual de un máximo del 25% de la base imponible del producto, imputándose anualmente como un gasto deducible.

Tabla 15: Presupuesto y relación de partidas presupuestarias. Moneda: euro.

PERSONAL					
APELLIDOS	**NOMBRE**	**CATEGORÍA**	**DEDICACIÓN**	**COSTE LABORAL**	**COSTE**
			HORAS	**PERSONA/HORAS**	**TOTAL (€)**
García Palao	Valerio	Proyectando	285,00	28,39	8.091,15
TOTAL PERSONAL			**285,00**		**8.091,15**
AMORTIZACIÓN (equipos)					
DESCRIPCIÓN	**COSTE (€)**	**% USO PROYECTO**	**DEDICACIÓN**	**PERIODO**	**COSTE**
			HORAS	**DEPRECIACIÓN**	**IMPUTABLE (€)**
Portátil proyectando	1.200,00	*100,00%*	285,00	48,00	237,50
TOTAL EQUIPOS			**285,00**		**237,50**
OTROS COSTES DIRECTOS					
DESCRIPCIÓN					**COSTE**
					IMPUTABLE (€)
Dietas	1 viaje a USJ ida y vuelta = 1 día = 3 comidas				75,00
Kilometraje	1 viaje a USJ ida y vuelta = 1 día = 465 Km * 2 * 0,25 €/km				232,50
Alojamiento	Sin alojamiento previsto				0,00
Materiales	Libros				50,00
Materiales	Impresión y encuadernación copia papel				66,25
Transporte	Envío copia impresa a USJ por mensajería				10,00
TOTAL OTROS COSTES DIRECTOS					**433,75**
RESUMEN					
PERSONAL					8.091,15
AMORTIZACIÓN (equipos)					237,50
OTROS COSTES DIRECTOS					433,75
TOTAL OTROS COSTES DIRECTOS					**8.762,40**
MARGEN BRUTO	*Costes indirectos (20%) + Beneficio (0%) =*			*20,00%*	*2.190,60*
TOTAL GENERAL					***10.953,00 €***

6.2. Beneficio económico.

Comentado en la partida de "margen bruto", el beneficio calificado a este proyecto es cero, puesto que no se pretende lucro con el mismo. Es un proyecto que demostrará el uso de la técnica de Topic Modeling y realizará una herramienta para demostrar su uso y que formará parte de otro sistema informático, estimándose que no hay beneficio económico por la venta o comercialización de los conocimientos o herramienta.

7. RESULTADOS:

En ese capítulo se relacionan los resultados asociados al finalizar el proyecto, así como pruebas realizadas, comentarios sobre errores detectados y herramientas (artefactos) producidos. Para finalizar se describen las desviaciones que ha sufrido el proyecto en su ciclo de vida.

7.1. Resultados obtenidos al finalizar el proyecto.

El esquema presentado en la Figura 24 muestra los principales resultados a la finalización del proyecto.

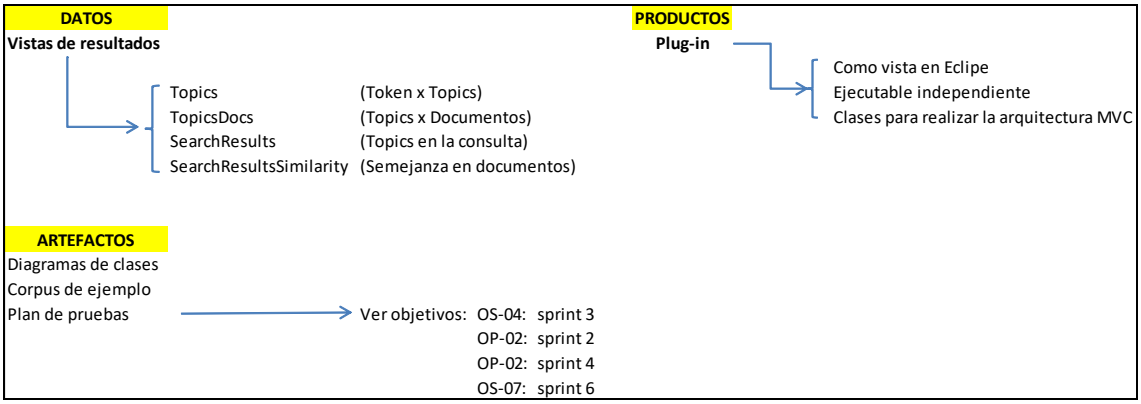


Figura 24: resultados al finalizar el proyecto.

Por una parte, se tiene el producto realizado. Como se ha descrito en otros capítulos, es una aplicación, de tipo plug-in, que contiene las herramientas y procedimientos para ejecutarse como vista, llamada "TopicModeling" en el IDE Eclipse y además posee una clase de tipo "main" para su ejecución sin necesidad de poner en marcha la instancia de Eclipse y activar la vista.

Asociado al producto se encuentran los datos. En la figura anterior se muestran los datos de salida, generados por las diferentes vistas que emite el plug-in. Dichas vistas se describen en el capítulo 5, aunque en la figura se muestra el valor de la información que contienen. Las dos últimas demuestran la utilidad de Topic Modeling para las búsquedas relacionadas en el análisis de software.

Para finalizar se encuentra la zona de artefactos que se han utilizado: diagramas de clases para elaborar el diseño y desarrollo, los corpus de ejemplos para realizar las pruebas y un plan de pruebas, asociado a tres objetivos del Product Backlog.

7.1.1. Plan de pruebas.

La metodología Scrum aporta la ventaja de comprobar el avance del producto en cada revisión del sprint, pudiendo introducir cambios o modificaciones para el siguiente sprint. Se trabaja con el producto mínimo viable para que el Product Owner pueda probarlo y comprobar

que los requerimientos, requisitos o funcionalidades están realizadas. Esta forma de avanzar ha permitido realizar diferentes pruebas conforme se ha avanzado en la ejecución del proyecto.

Como se detalla en la Figura 24, el plan de pruebas se realiza durante la consecución de los objetivos OS-04, OP-02 y OS-07. Además, para su ejecución se contemplan los escenarios previstos en el objetivo OS-06, escenarios hash (ID-6.00) y ficheros Java (ID-6,01), sobre ellos se evaluará el plan según OS-07.

Tabla 16: Plan de pruebas realizado.

OBJETIVO ID	OBJETIVO DESCRIPCIÓN	SPRINT ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES ID	REQUISITOS, REQUERIMIENTOS Y FUNCIONALIDADES DESCRIPCIÓN	TIPO
OP-02	**Implementar una aplicación en Java que utilice la técnica Topic Modeling para buscar términos relacionados**				
		2			
			2,12	Pruebas sobre ficheros JAVA y clase tm (Topic Modeling)	Pruebas
			2,13	Estudiar diferentes métodos de MALLET para realizar TM y elegir el más adecuado	Pruebas
		4			
			2,14	Validar las pruebas con programa TM "main"	Pruebas
			2,22	Crear un plug-in de ejemplo	Pruebas
OS-04	**Investigar y estudiar la técnica y herramienta para su uso en Topic Modeling.**				
		3			
			4,12	Pruebas con la herramienta seleccionada para determinar uso en TM	Diseño
			4,13	Pruebas específicas con la herramienta seleccionada para determinar uso en TM asociado al Análisis de Software. Ver uso de API o librerías en lenguaje Java	Diseño
			4,14	Documentar pruebas para presentar a Dirección Proyecto	Diseño
OS-07	**Evaluar comportamiento de Topic Modeling en el conjunto de datos del corpus seleccionado para ver resultados y establecer conclusiones**				
		6			
			7,01	Tipo prueba 1: Datos HashMap y cadena de búsqueda	Salida
			7,02	Tipo prueba 2: Fichero Java y cadena de búsqueda	Salida
			7,03	Tipo prueba 3: Datos Java y alteración de parámetros	Salida
			7,04	Tipo prueba 4: Datos HashMap y Java y cadenas de búsqueda.	Salida
			7,05	Tipo prueba 5: alterar cadenas de búsqueda dentro del mismo modelo	Salida

La Tabla 16 presenta el plan de pruebas confeccionado y realizado en el proyecto. En el capítulo 5, se comentan y plasman las pruebas realizadas en los objetivos siguientes:

- OS-04:
  - Sprint 3, con identificadores 4,12, 4,13 y 4,14 que corresponden con los puntos del capítulo 5.2.2 apartado *Sprint 3* y representado en Figura 9.
- OP-02:
  - Sprint 2, con identificadores 2,12 y 2,13 que corresponden con los puntos del capítulo 5.6.2 apartado *Sprint 2* y representado en Figura 15.
  - Sprint 4, con identificadores 2,14 y 2,22 que corresponden con los puntos del capítulo 5.6.3 apartado *Sprint 4* y representado en Figura 15 y Figura 16.

Quedando para este capítulo la exposición de la batería de pruebas realizada para el objetivo OS-07. Dichas pruebas contemplan los escenarios previstos en el objetivo OS-06.

Tipo prueba 1: datos HashMap y cadena de búsqueda.

Se evalúa el corpus de datos tipo hash contra una cadena de búsqueda de tres palabras. Los parámetros asociados al proceso de modelaje son los que se muestran por defecto en la vista inicial del plug-in llamada "Setup". Se han marcado en color verde los valores por defecto y en color rojo el campo dónde se introduce la cadena de términos a buscar. Ver Figura 25.

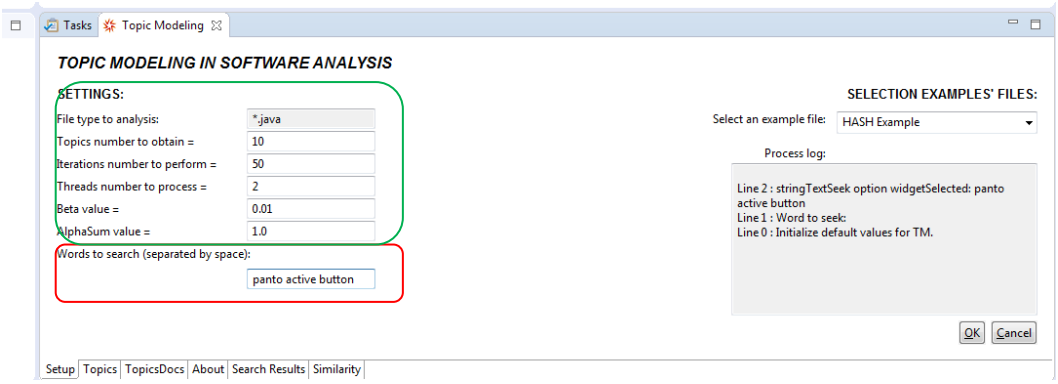


Figura 25: Vista "Topic Modeling". Solapa "Setup". Valores por defecto para Prueba 1.

La Figura 26, muestra el resultado de la ejecución del proceso, y en color azul la máxima probabilidad de contener las 3 palabras de la cadena de búsqueda en el documento señalado.

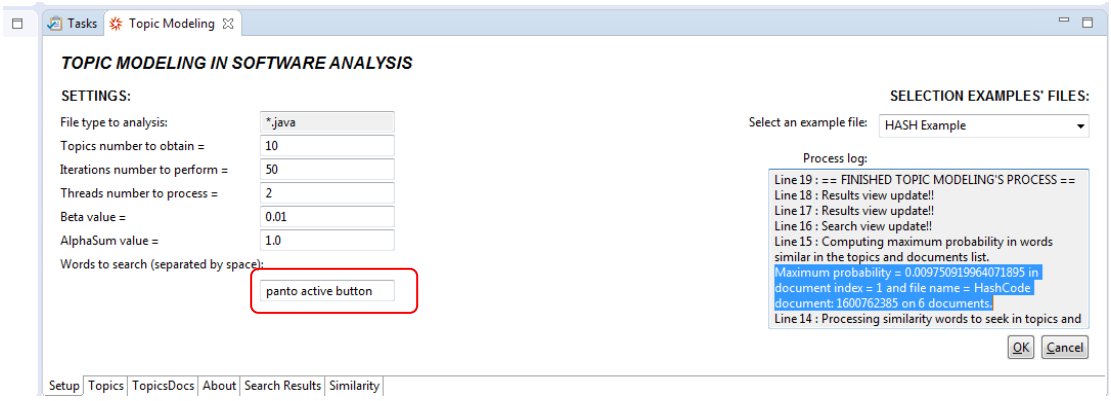


Figura 26: Vista "Topic Modeling". Solapa "Setup". Datos en campo "log" en Prueba 1.

La Figura 22 y Figura 23, asociadas al capítulo 5, muestran los resultados esperados. Figura 23 confirma que el "*documento 1*" es el que más probabilidad tiene de contener la secuencia de palabras "*panto active button*". Es una probabilidad muy pequeña, cerca del 1%, pero es el que mayor probabilidad tiene. Puede examinarse la Figura 27 y comprobar que realmente ese documento contiene las tres palabras de la cadena. Esta prueba confirma la bondad de Topic Modeling para localizar documentos que contengan los términos en base a los términos de la consulta.

La Figura 22 muestra la vista dónde la búsqueda, en vez de realizarse agrupando los términos de la consulta, se ha realizado contra la lista de topics y sus palabras, identificando los que la contienen y mostrando sus datos y los documentos asociados a cada topic. Se indica la probabilidad de cada palabra en el topic y a la vez la probabilidad del topic en cada documento. Es decir, muestra la matriz fi y zeta en una sola vista, siendo por tanto complementaria a la anterior vista en Figura 23.

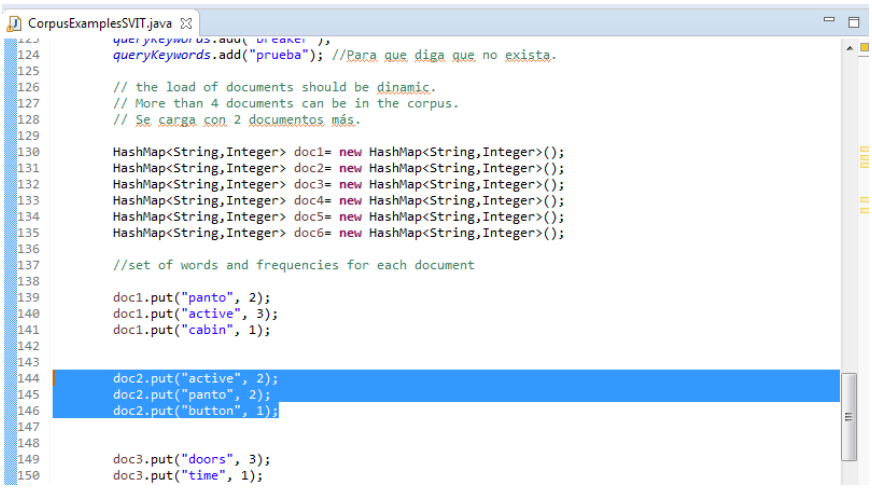


Figura 27: Inicialización del documento 2 del corpus tipo hash.

Tipo prueba 2: fichero Java y cadena de búsqueda.

La siguiente prueba es similar a la anterior, pero cambiando el tipo de fichero de ejemplo y la cadena de búsqueda, así el tipo de fichero a elegir es "JAVA Example: BASIC" y la cadena de palabras a buscar es "topic alphabet instance".

La Figura 28 muestra la semejanza de documentos. Encontrando que el documento número 3 es el que más probabilidad tiene de alojar la cadena de búsqueda, aunque tenga una probabilidad con número de 10^{-6} que aparentemente es baja. La Figura 29 corrobora que las tres palabras de la cadena de búsqueda no se encuentran en el mismo topic, justificando, en parte ese valor de probabilidad tan bajo.

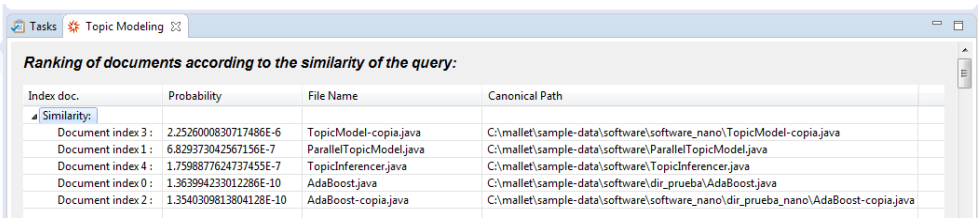


Figura 28: Vista "Topic Modeling". Solapa "Similarity". Resultado tras en Prueba 2.

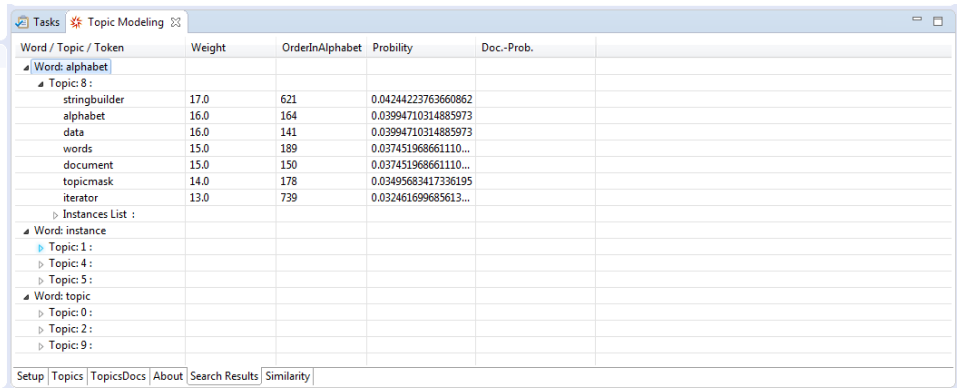
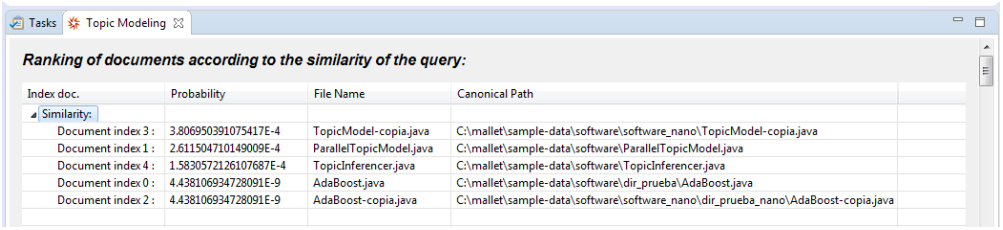


Figura 29: Vista "Topic Modeling". Solapa "SearchResult". Resultado en Prueba 2.

Tipo prueba 3: Datos Java y alteración de parámetros.

La siguiente prueba es similar a la anterior, pero cambiando los parámetros del número de iteraciones, de 50 a 100, y el número de topics de 10 a 5. Con el objetivo de mostrar la influencia de estos parámetros en el resultado sobre el mismo corpus seleccionado en Prueba 2. La Figura 30, muestra el resultado tras la ejecución. Se observa que la probabilidad ha subido bastante, pasando de ser un valor de 10^{-6} a 10^{-4} y señalándola sobre el mismo documento 3.

Destaca en la Figura 31 que la palabra "instance", en esta ejecución, no tiene topic asociado. Una de las características de Topic Modeling y de los procesos de muestreo que utiliza, es que los resultados en ejecuciones secuenciadas no son iguales.

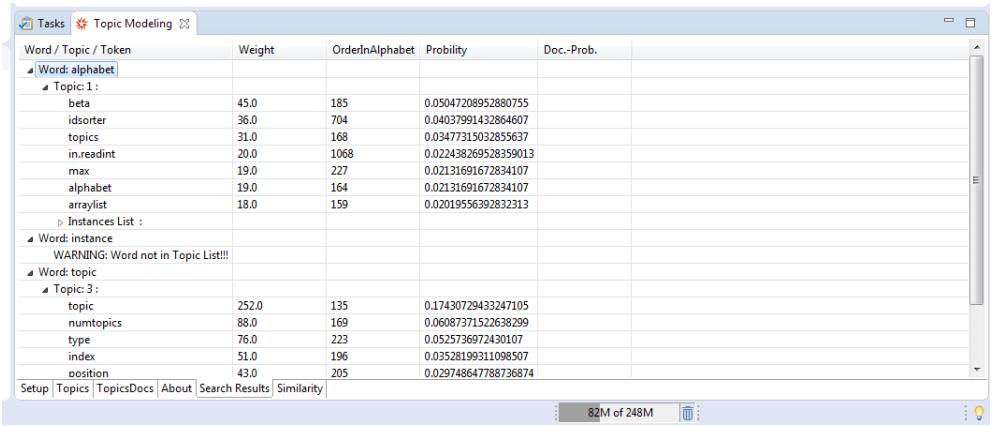

Tasks | Topic Modeling

Ranking of documents according to the similarity of the query:

Index doc.	Probability	File Name	Canonical Path
Similarity:			
Document index 3 :	3.806950391075417E-4	TopicModel-copia.java	C:\mallet\sample-data\software\software_nano\TopicModel-copia.java
Document index 1 :	2.6115047101490O9E-4	ParallelTopicModel.java	C:\mallet\sample-data\software\ParallelTopicModel.java
Document index 4 :	1.5830572126107687E-4	TopicInferencer.java	C:\mallet\sample-data\software\TopicInferencer.java
Document index 0 :	4.438106934728091E-9	AdaBoost.java	C:\mallet\sample-data\software\dir_prueba\AdaBoost.java
Document index 2 :	4.438106934728091E-9	AdaBoost-copia.java	C:\mallet\sample-data\software\software_nano\dir_prueba_nano\AdaBoost-copia.java

Figura 30: Vista "Topic Modeling". Solapa "Similarity". Resultado en Prueba 3.

En este caso ha sido así. Se ha realizado la prueba otra vez, y esta vez es la palabra "alphabet" quien no tiene topic asociado, pero si "instance". Ver Figura 32.

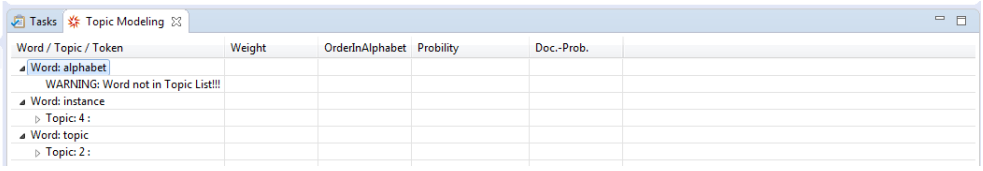

Tasks | Topic Modeling

Word / Topic / Token	Weight	OrderInAlphabet	Probility	Doc.-Prob.
Word: alphabet				
Topic: 1 :				
beta	45.0	185	0.05047208952880755	
idsorter	36.0	704	0.040379914328646O7	
topics	31.0	168	0.034773150328556З7	
in.readint	20.0	1068	0.022438269528359013	
max	19.0	227	0.021316916728341O7	
alphabet	19.0	164	0.021316916728341O7	
arraylist	18.0	159	0.020195563928323l3	
Instances List :				
Word: instance				
WARNING: Word not in Topic List!!!				
Word: topic				
Topic: 3 :				
topic	252.0	135	0.17430729433247105	
numtopics	88.0	169	0.06087371522638299	
type	76.0	223	0.0525736972430107	
index	51.0	196	0.035281993110985O7	
position	43.0	205	0.029748647788736874	

Setup | Topics | TopicsDocs | About | Search Results | Similarity

82M of 248M

Figura 31: Vista "Topic Modeling". Solapa "SearchResult". Resultado en Prueba 3.


Tasks | Topic Modeling

Word / Topic / Token	Weight	OrderInAlphabet	Probility	Doc.-Prob.
Word: alphabet				
WARNING: Word not in Topic List!!!				
Word: instance				
Topic: 4 :				
Word: topic				
Topic: 2 :				

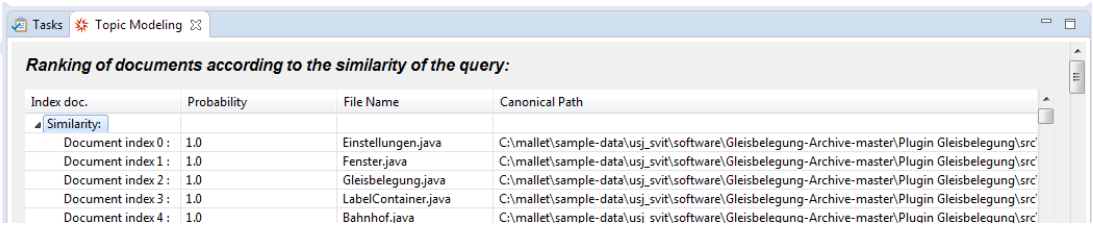
Figura 32: Solapa "SearchResult" tras volver a ejecutar Prueba 3.

Tipo prueba 4: Datos HashMap y Java y cadenas de búsqueda.

En esta prueba el objetivo es demostrar que la aplicación puede trabajar con grandes cantidades de ficheros Java. A nivel de parámetros es igual a la anterior: se mantienen los parámetros del número de iteraciones en 100, el número de topics en 5 y manteniendo la

misma cadena de búsqueda de la Prueba 3. Sólo se cambia, en la "selección de ejemplos", a la opción "JAVA Example: ALL of the above".

En la Figura 33, pestaña "Similarity", se exponen los resultados tras la ejecución. No han sido buenos, al dar valor de probabilidad = 1 en todos los documentos. Esto significa que no ha encontrado ningún documento con una probabilidad suficiente que contenga las tres palabras de la cadena de búsqueda. Como dato de interés de la prueba, la aplicación ha examinado todos los documentos en 1,9861 minutos. Dando como resultado positivo que cantidades considerables de ficheros son admitidas por la aplicación sin problemas.

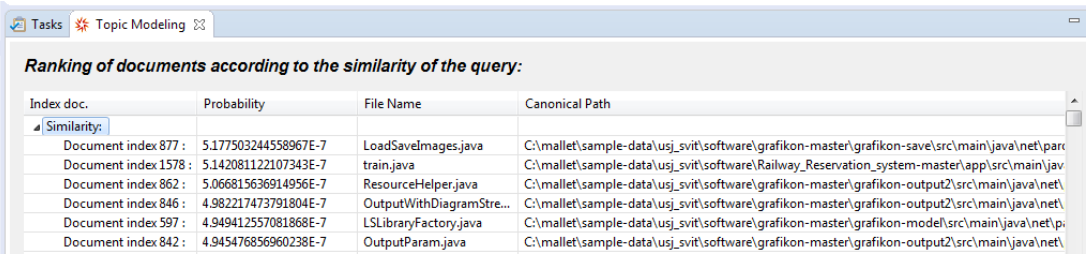

Tasks | Topic Modeling

Ranking of documents according to the similarity of the query:

Index doc.	Probability	File Name	Canonical Path
Similarity:			
Document index 0 :	1.0	Einstellungen.java	C:\mallet\sample-data\usj_svit\software\Gleisbelegung-Archive-master\Plugin Gleisbelegung\src
Document index 1 :	1.0	Fenster.java	C:\mallet\sample-data\usj_svit\software\Gleisbelegung-Archive-master\Plugin Gleisbelegung\src
Document index 2 :	1.0	Gleisbelegung.java	C:\mallet\sample-data\usj_svit\software\Gleisbelegung-Archive-master\Plugin Gleisbelegung\src
Document index 3 :	1.0	LabelContainer.java	C:\mallet\sample-data\usj_svit\software\Gleisbelegung-Archive-master\Plugin Gleisbelegung\src
Document index 4 :	1.0	Bahnhof.java	C:\mallet\sample-data\usj_svit\software\Gleisbelegung-Archive-master\Plugin Gleisbelegung\src

Figura 33: Vista "Topic Modeling". Solapa "Similarity". Resultado en Prueba 4.

En la Figura 34 se muestran los resultados alterando la cadena de búsqueda e introduciendo una diferente: "string active rail train". Puesto que al construir el corpus se seleccionaron ficheros de proyectos sobre trenes, se añaden dos palabras con referencia a ellos para "influir" en el éxito de la búsqueda. El resultado es positivo y ha encontrado más de un documento con una probabilidad de contener las cuatro palabras. Aunque la probabilidad es un valor numérico de 10^{-7} existen documentos que pueden contenerla.


Tasks | Topic Modeling

Ranking of documents according to the similarity of the query:

Index doc.	Probability	File Name	Canonical Path
Similarity:			
Document index 877 :	5.1775032445589 67E-7	LoadSaveImages.java	C:\mallet\sample-data\usj_svit\software\grafikon-master\grafikon-save\src\main\java\net\par
Document index 1578 :	5.1420811221073 43E-7	train.java	C:\mallet\sample-data\usj_svit\software\Railway_Reservation_system-master\app\src\main\jav
Document index 862 :	5.0668156369145 56E-7	ResourceHelper.java	C:\mallet\sample-data\usj_svit\software\grafikon-master\grafikon-output2\src\main\java\net\
Document index 846 :	4.9822174737918 04E-7	OutputWithDiagramStre...	C:\mallet\sample-data\usj_svit\software\grafikon-master\grafikon-output2\src\main\java\net\
Document index 597 :	4.9494125570818 68E-7	LSLibraryFactory.java	C:\mallet\sample-data\usj_svit\software\grafikon-master\grafikon-model\src\main\java\net\p
Document index 842 :	4.9454768569602 38E-7	OutputParam.java	C:\mallet\sample-data\usj_svit\software\grafikon-master\grafikon-output2\src\main\java\net\

Figura 34: "Topic Modeling". Solapa "Similarity". Resultado en Prueba 4 con otra cadena de búsqueda.

La ejecución de esta prueba ha tarda menos que la anterior, concretamente 0,7677 minutos, a pesar de ejecutar el proceso completo. Esto es debido a la gestión de memoria y de objetos que maneja Java, junto con la carga del procesador y disco del ordenador en ese momento.

Tipo prueba 5: alterar cadenas de búsqueda dentro del mismo modelo.

Esta prueba se diseña para comprobar la agilidad de las búsquedas en el caso de tener el mismo modelo. En la revisión del sprint 5 y planificación el sprint 6, se acuerda introducir una opción para repetir la consulta sin necesidad de volver a modelar, con el objetivo de acelerar las consultas. Esto se referencia con ID-2,92 e ID-2,93.

La Figura 35 presenta la vista "Setup" con un botón llamado "Query" justo al lado de la casilla de términos a buscar. Éste sólo se activa una vez que se ha realizado el modelado. Esto no es óbice para que la primera consulta, con el botón desactivado, se pueda realizar tras el modelado. Segundas y sucesivas se realizan ya con la pulsación del botón. La Figura 36 muestra el resultado tras la consulta de la palabra "train" una vez ejecutado el modelo con la opción "ALL of the above".

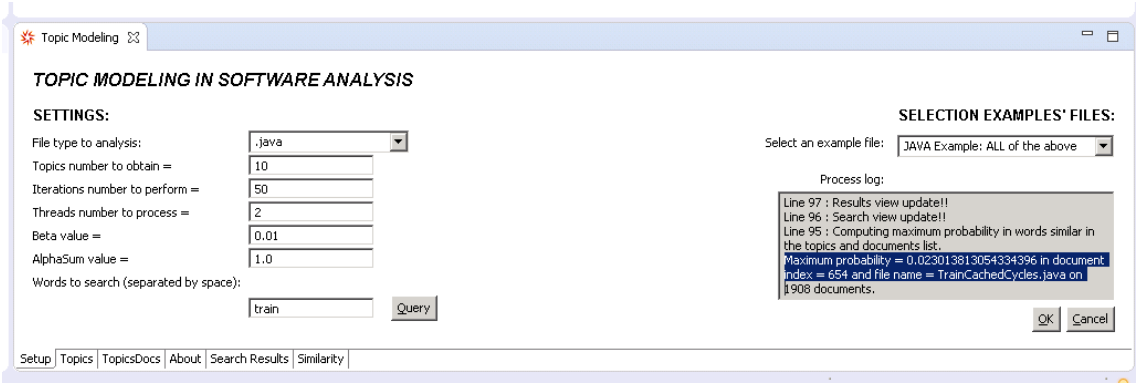


Figura 35: "Topic Modeling". Solapa "Setup" con botón "query" incorporado en Prueba 5.

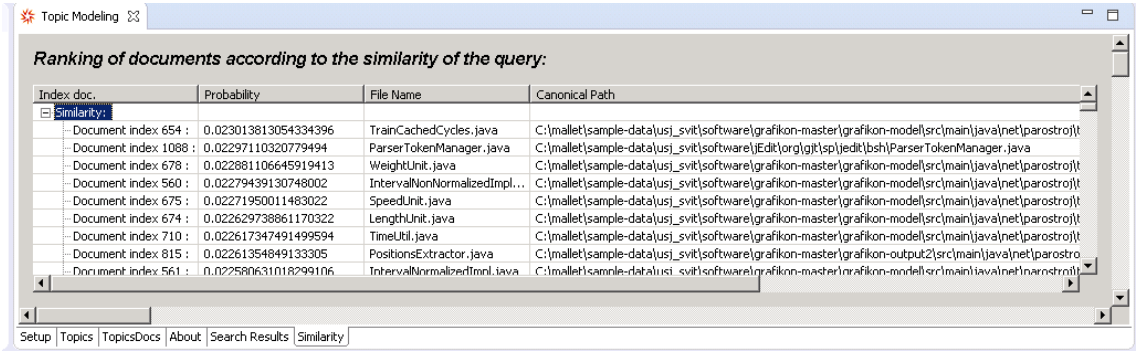


Figura 36: "Topic Modeling". Solapa "Similarity". Resultado en Prueba 5.

7.1.2. Conclusiones del plan de pruebas.

En esta subsección se enumeran las conclusiones obtenidas de las diferentes pruebas realizadas. Son las siguientes:

i) Topic Modeling se aplica correctamente sobre ficheros y estructura hash conteniendo texto.

ii) El orden es importante e importa. Es decir, importa cómo lee los datos el proceso, los convierte en tokens, la frecuencia de estos y su coocurrencia. Es importante señalar que se trabaja con textos, que acaban convertidos en valores numéricos.

iii) El resultado sale diferente en cada ejecución. Es un procedimiento de cálculo estadístico. Así que los ejemplos mostrados no coincidirán con los de otras ejecuciones.

iv) La salida, topics y sus componentes tokens, pertenecen al conjunto de documentos y no a ellos individualmente. Se expresa en valor de probabilidad la pertenencia de ese topic al documento y del token al topic, mediante las dos matrices comentadas y la estructura de datos y métodos desarrollados para ello.

v) En el capítulo 5 se recomiendan, según McCallum (39) un número determinado de iteraciones según sean pruebas o no. Cuantas más iteraciones (sin retoque de los parámetros) el modelo mejora la probabilidad de los tokens, pero a partir de ciertas iteraciones, la mejora es mínima, llegando al tercer decimal de la probabilidad.

vi) No hay etiqueta para el topic generada automáticamente, al ser Topic Modeling una técnica no supervisada. La estructura de datos DataFile está preparada para ello, aunque no se hace uso de ella en la aplicación.

vii) Importante es realizar ajustes con las stopwords o palabras vacías para no tener token del tipo: *string, file, param, program*, etc. o palabras unidas o cortadas: *Stringof, System.out.println(", gridbagconstraints*, etc. se ven reflejadas en los topics.

viii) Una dispersión de palabras por los topics y en los documentos, hacen que las probabilidades se reduzcan. Éste es uno de los motivos que generó la utilización de dos sistemas de búsquedas (pestañas "SearchResults" y "Similarity") complementarios entre sí.

No se tiene errores significativos en las pruebas excepto errores de dependencias de los paquetes utilizados en el entorno del plug-in; solucionados con las modificaciones pertinentes en los ficheros "manifest" o "build". También aparecen errores de "OutOfMemoryError" o de "NullPointer" al procesar muchos tokens o utilización de algoritmos con orden complejidad elevada, o acceso a valores de listas, vectores o matrices fuera de rango y fueron solucionados poniendo límites máximos y mínimos en muchos de los objetos a procesar.

7.2. Desviaciones aparecidas en el proyecto.

En esta sección se describen las diferentes desviaciones detectadas en el ciclo de vida del proyecto. Se detallan desviaciones asociadas a objetivos, metodología y económicas.

7.2.1. Objetivos.

En el área de los objetivos, descritos en el capítulo 3, no ha habido desviaciones de ningún tipo. El empleo de Scrum ha posibilitado ir creciendo en las tareas de cada objetivo sin necesidad de alterar ninguno de los descritos en el Product Backlog y cerrando cada uno de ellos satisfactoriamente.

7.2.2. Metodología

El área de la metodología, descrita en el capítulo 4, ha sufrido desviaciones que se presentan en esta subsección. La Tabla 18 o "Final Sprint Backlog" refleja los sprints tras la ejecución del proyecto. Utilizar Scrum ha permitido hacer cambios durante el ciclo de vida del proyecto. Las diferencias entre lo planificado y lo ocurrido durante el transcurso del proyecto son las siguientes:

- *Diferencia en los sprints.* Si bien el número de sprints se mantiene se han alterado las condiciones de satisfacción o "release" en cada sprint. En el sprint de lanzamiento (sprint 0) ya se acuerda con Product Owner intentar concentrar los esfuerzos durante los primeros sprints. Dejando los tres últimos sprints para los objetivos OS-10 y OS-11.
- *Concentración de esfuerzos en los sprints.* La distribución inicial, que rozaba casi un reparto equitativo entre sprints, lo que se conoce como "timeboxing" o fijar un tiempo máximo para conseguir objetivos, se alteró desde el principio por el motivo de desconocer la técnica (obligación, por tanto, de investigar, estudiar y emplear más tiempo), elección de herramientas, falta de documentación asociada a la misma y a la creación de plug-in. Así se plantean, inicialmente, sprints más largos, mayores de un mes de duración; estableciendo en cada revisión qué mostrar en el siguiente sprint, duración del mismo y esfuerzo en puntos y horas de trabajo, para ajustarse a fecha.
- *Empleo de más horas.* Debido a la envergadura que el proyecto iba tomando, a nivel de investigación y estudio en los objetivos OS-01 a OS-04. El resto de exceso de horas se justifica por:
    - Dificultades en plug-in: a la hora del uso de "widgets" en el plug-in, la falta de documentación para su creación, su representación en las vistas, disposición de componentes, errores de la generación de la instancia y gestión de listeners.
    - Capítulos 2 y 5: Se consumen más horas de las estimadas en los desarrollos de los capítulos y realización de memoria.
    - Análisis de la herramienta MALLET. El compendio de todas sus librerías y uso tiene una falta de documentación grave, que generó desviaciones de horas.

Tabla 17: resumen de desviaciones sufridas durante el proyecto en indicadores principales.

UD	FINAL SPRINT BACKLOG										Total
Medida	0	1	2	3	4	5	6	7	8	9	
Nº semanas trabajo estimadas	3,00	4,00	4,00	4,00	4,00	4,00	4,00	4,00	4,00	4,00	39,00
Nº semanas trabajo reales	7,30	5,60	11,90	5,90	1,90	2,00	5,00	0,90	0,70	2,70	43,90
Diferencia real-estimada	*4,30*	*1,60*	*7,90*	*1,90*	*-2,10*	*-2,00*	*1,00*	*-3,10*	*-3,30*	*-1,30*	*4,90*
Nº días trabajo estimado	18,00	25,00	25,00	25,00	24,00	25,00	25,00	25,00	25,00	25,00	242,00
Nº días trabajo reales	51,00	39,00	83,00	41,00	13,00	14,00	35,00	6,00	5,00	19,00	306,00
Diferencia real-estimada	*33,00*	*14,00*	*58,00*	*16,00*	*-11,00*	*-11,00*	*10,00*	*-19,00*	*-20,00*	*-6,00*	*64,00*
Nº horas trabajo estimado	21,20	29,44	29,44	29,44	28,26	29,44	29,44	29,44	29,44	29,44	284,98
Nº horas trabajo reales	60,06	45,93	97,75	48,29	52,00	59,50	148,75	22,38	5,89	7,07	547,62
Diferencia real-estimada	*38,86*	*16,49*	*68,31*	*18,85*	*23,74*	*30,06*	*119,31*	*-7,06*	*-23,55*	*-22,37*	*262,64*
Nº puntos esfuerzo estimado	*21,00*	*59,00*	*90,00*	*82,00*	*79,00*	*77,00*	*85,00*	*85,00*	*72,00*	*15,00*	665,00
Nº puntos esfuerzo reales	*39,00*	*49,00*	*167,00*	*112,00*	*75,00*	*99,00*	*100,00*	*23,00*	*5,00*	*3,00*	672,00
Diferencia real-estimada	*18,00*	*-10,00*	*77,00*	*30,00*	*-4,00*	*22,00*	*15,00*	*-62,00*	*-67,00*	*-12,00*	*7,00*

La Tabla 17 muestra las desviaciones de los indicadores principales, como son semanas y días de trabajo, horas trabajadas y puntos de esfuerzo. El dato más importante es la desviación en las horas consumidas de trabajo. Ha implicado de pasar de una ratio de horas/día de 1,17769 a 1,78961 y en días el aumento ha sido 64 días.

A nivel de esfuerzos sólo ha habido una pequeña desviación de 7 puntos que es generada por contemplar los esfuerzos de administración y gestión de la metodología Scrum en el proyecto: reuniones de revisión y planificación con Product Owner (incluye tutorías con dirección de proyecto), gestión de tareas y documentación, junto con un aumento de esfuerzo en el objetivo OS-10 que pasa de 233 a 241 puntos.

La Figura 37 muestra las gráficas de horas y esfuerzo comparando la estimación con la realidad del proyecto. En la parte izquierda aparece la comparativa entre horas estimadas vs reales. Se observa que no hay una continuidad debido a la carga de trabajo que sufre un desequilibrio, por los motivos antes expuestos, en los sprints 2 y 6. La ventaja de ese empleo de horas supone, que el proyecto avanzó bastante rápido pudiendo prácticamente estar terminado en el sprint 6 ó 7.

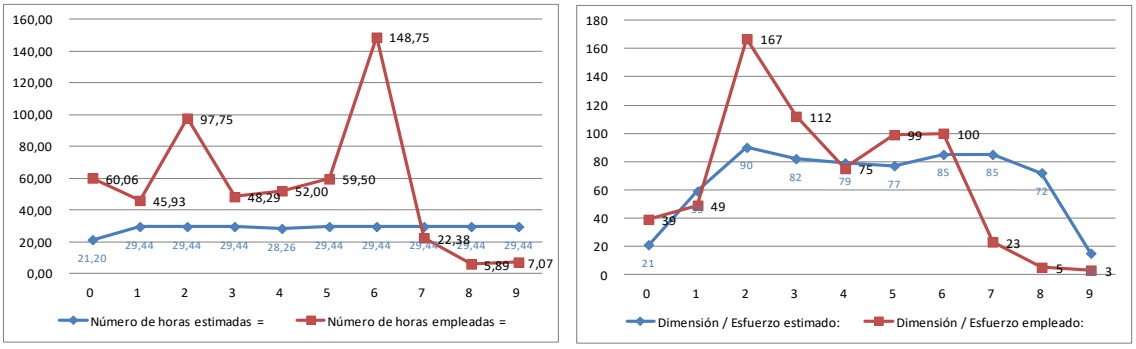


Figura 37: gráficas de horas y esfuerzo estimados y reales del proyecto.

Tabla 18: Lista de tareas final de la iteración o Final Sprint Backlog.

Objetivo ID	0	1	2	3	FINAL SPRINT BACKLOG 4	5	6	7	8	9	Total x ID	CAPÍTULO ASOCIADO
OS-08	21										21	4
OS-01	13	21									34	2
OS-02		21	34								55	2
OS-03			21	34							55	2
OS-04			13	21							34	2
OS-09		5									5	2
OP-01			13			8					21	5
OS-05			8		5						13	5
OP-02			55		13	55	21				144	5
OS-06							1				1	5 y 7
OS-07							21				21	7
OS-10				55	55	34	55	21			241	memoria
OS-11									5	3	8	defensa
OS-12	5	2	2	2	2	2	2	2	0	0	19	admón
Total x ID	**39**	**49**	**167**	**112**	**75**	**99**	**100**	**23**	**5**	**3**	**672**	

	De:	09-sep-19	01-nov-19	11-dic-19	04-mar-20	15-abr-20	29-abr-20	14-may-20	19-jun-20	25-jun-20	01-jul-20	
Revisión Sprint =	A:	**30-oct-19**	**10-dic-19**	**03-mar-20**	**14-abr-20**	**28-abr-20**	**13-may-20**	18-jun-20	**25-jun-20**	**30-jun-20**	**20-jul-20**	
Nombre sprint:		Lanz.+Est (I) Matr+Est. Arte	Estudio (I) Est. Arte+TM-AS	Est. (II)+ Apli (I) Relación TM-AS	Memoria (I) Relación TM-AS	Aplicación (II) TM y Apli.	Aplicación (III) Apli. Y Cap.	Memoria (II) Capítulos	Cierre proyecto Entrega proy.	Const. Trib. Calend. Def.	Presentación Defensa PFG	
Release o condiciones de satisfacción:												
Dimensión / Esfuerzo empleado:	672	39	49	167	112	75	99	100	23	5	3	672
Velocidad del equipo (puntos/semana):	15,31	5,34	8,75	14,03	18,98	39,47	49,50	20,00	25,56	7,14	1,11	
Número de semanas (7 días) =	44	7,3	5,6	11,9	5,9	1,9	2,0	5,0	0,9	0,7	2,7	
Número de días =	306	51	39	83	41	13	14	35	6	5	19	*Ratio h/día*
Número de horas empleadas =	547,62	60,06	45,93	97,75	48,29	52,00	59,50	148,75	22,38	5,89	7,07	1,78961
Valor de quemado conseguido =	672,00	633	584	417	305	230	131	31	8	3	0	

Núm. de horas estimadas TOTALES Guía Docente =	300,00
Núm. horas estimadas Trabajo personal en ProyectoHoras según Guía Docente =	285,00
Reuniones de seguimiento con el Director/Tutor (10 horas máximo =	10,00
Actividades de evaluación =	1,00
Otras actividades prácticas =	4,00

7.2.3. Económicas.

La valoración económica final del proyecto se detalla en la Tabla 19, dónde se observan las siguientes desviaciones económicas frente al presupuesto:

i) El apartado de personal muestra la desviación más importante, que es el coste total de horas empleadas. Esta es debida, como se menciona en la subsección anterior al incremento de horas que ha sufrido el proyecto.

Tabla 19: facturación vs presupuesto coste proyecto.

PERSONAL							
APELLIDOS	**NOMBRE**	**CATEGORÍA**	**DEDICACIÓN HORAS**	**COSTE LABORAL PERSONA/HORAS**	**COSTE TOTAL (€)**	***PRESUPUESTO TOTAL (€)***	***DIFERENCIA***
García Palao	Valerio	Proyectando	547,62	28,39	15.546,93	*8.091,15*	*-7.455,78*
TOTAL PERSONAL			**547,62**		**15.546,93**	***8.091,15***	***-7.455,78***
AMORTIZACIÓN (equipos)							
DESCRIPCIÓN	**COSTE (€)**	**% USO PROYECTO**	**DEDICACIÓN HORAS**	**PERIODO DEPRECIACIÓN**	**COSTE IMPUTABLE (€)**	***PRESUPUESTO IMPUTABLE (€)***	***DIFERENCIA***
Portátil proyectando	1.200,00	*100,00%*	547,62	48,00	456,35	*237,50*	*-218,85*
TOTAL EQUIPOS			**547,62**		**456,35**	***237,50***	***-218,85***
OTROS COSTES DIRECTOS							
DESCRIPCIÓN					**COSTE IMPUTABLE (€)**	***PRESUPUESTO IMPUTABLE (€)***	***DIFERENCIA***
Dietas					0,00	*75,00*	*75,00*
Kilometraje					0,00	*232,50*	*232,50*
Alojamiento					0,00	*0,00*	*0,00*
Materiales	Libros				50,00	*50,00*	*0,00*
Materiales	Impresión y encuadernación copia papel				66,25	*66,25*	*0,00*
Transporte	Envío copia impresa a USJ por mensajería				10,00	*10,00*	*0,00*
TOTAL OTROS COSTES DIRECTOS					**126,25**	***433,75***	***307,50***
RESUMEN COSTES DIRECTOS							
PERSONAL					15.546,93	*8.091,15*	*-7.455,78*
AMORTIZACIÓN (equipos)					456,35	*237,50*	*-218,85*
OTROS COSTES DIRECTOS					126,25	*433,75*	*307,50*
SUBTOTAL COSTES DIRECTOS					**16.129,53**	***8.762,40***	***-7.367,13***
MARGEN BRUTO	*Costes indirectos (20%) + Beneficio (0%) =*			*20,00%*	*4.032,38*	*2.190,60*	*-1.841,78*
TOTAL GENERAL					***20.161,91 €***	***10.953,00 €***	***-9.208,91***

ii) En el apartado de amortización existe una desviación derivada del incremento de horas. Al utilizar más horas el portátil del proyectando, se ha generado un aumento también de la amortización.

iii) En el apartado de margen bruto, al ser un porcentaje sobre los costes directos y estos variar, se genera una nueva desviación.

En resumen, las variaciones antes comentadas generan un aumento sobre el presupuesto inicial de 9.208,91 euros, generando un total en la factura de 20.161,91 euros impuestos excluidos.

8. CONCLUSIONES:

Este capítulo presenta las conclusiones al finalizar el proyecto, indicando las posibles mejoras y ampliaciones. El proyecto tiene como destino lo indicado en su título, es decir, utilizar la técnica de Topic Modeling para realizar consultas compuestas y buscar términos relacionados en documentos, indicando cuáles de ellos pueden contenerlos. Profundiza en uno de los campos de aplicación de la ciencia de datos como es la minería de textos y el procesamiento del lenguaje natural, llegando hasta el análisis de textos de código de software. López-Nozal et al. (20) ya describen Topic Modeling como una técnica bastante relevante en la aplicación de las mejoras sobre los procesos del ciclo de vida del software y Linstead et al. (19) proponen su uso para resolver problemas de análisis de software en la ingeniería del software.

Para la realización se dispone de los objetivos primarios y secundarios, según se describe en el Product Backlog, con un cumplimiento de todos ellos durante el desarrollo del proyecto, como se demostró en los capítulos 5 y 7. Se ha empleado la metodología Scrum como herramienta de organización, gestión del proyecto y como medio para alcanzar la finalización de los objetivos realizando revisiones continuadas.

Se ha investigado y estudiado, en la medida del tiempo disponible y objetivos alcanzables, la ciencia de datos y gran parte de sus áreas de trabajo, aprendiendo la relación entre todas ellas, hasta llegar a observar que LDA es el algoritmo de referencia en Topic Modeling, seleccionado una herramienta como MALLET para realizar la implementación de la aplicación en forma de plug-in. Linstead et al. (19) evidencian que los algoritmos de modelado como LDA proporcionan una herramienta estadística estable y escalable para la comprensión, complejidad y evolución que tiene el software.

Se ha trabajado en la creación del corpus ejemplo y clases para su importación y procesamiento. Los ejemplos se han obtenido de los repositorios abiertos de Internet seleccionando aplicaciones completas, para que la aplicación pudiera discriminar los ficheros no deseados. Como complemento se prepara la aplicación para la obtención de datos por estructuras de datos de tipo hash. Esto hace de las librerías creadas, herramientas para la integración en sistemas mayores.

Por último, se desarrolló una aplicación, de tipo plug-in, que sirvió para visualizar la utilidad del modelo y herramienta elegida en el diseño y desarrollo del mismo. En ella se integra el proceso de modelado y su proceso anterior de procesamiento del texto en bruto, hasta tener una "bag of words" adecuada para el modelado. Para probar la aplicación se definieron dos escenarios dónde se vio el comportamiento de la aplicación y técnica. Se ha comprobado que LDA es una buena herramienta para descubrir grupos latentes de palabras que describen los topics (temas) en un texto.

Se ha verificado, con dos sistemas de búsqueda relacionadas, mostradas en las vistas "SearchResults" y "Similarity", que cumplen con el objetivo del proyecto y además añaden valor extra: creando una salida que puede servir de entrada a otras técnicas de aprendizaje automático o a otros sistemas mayores de software.

Como conclusión, la realización del proyecto ha servido para dar a conocer una nueva perspectiva de una parte de la ciencia de datos: la aplicación de Topic Modeling y cómo se puede integrar esta herramienta en otros campos de la ciencia. Existen estudios con una serie de aplicaciones, relacionadas con la ingeniería del software como son análisis previo al diseño, mantenimiento, corrección de bugs o errores, etc. Pero la técnica puede ir más allá e integrarse, por ejemplo, en sistemas o aplicaciones relacionadas con las nuevas técnicas de "low-code" o "no-code" para la realización de software.

8.1. Mejoras y futuras ampliaciones.

En este apartado se relacionan las mejoras y ampliaciones que pueden realizarse sobre los objetivos del proyecto, que por cuestión de amplitud y esfuerzo no se han llegado a ejecutar.

i) A nivel de la aplicación desarrollada:
   a. El guardado y la carga del modelo generado.
   b. Mejorar funcionalidad con botones y visualización proceso de cálculo.
   c. Ampliar el procesamiento de las entradas: nuevas formas introducir el alfabeto.
   d. Implementar acciones de uso: localizar las palabras de la búsqueda en el texto, marcar las palabras de los diferentes topics, etc.
   e. Preparar el plug-in para cargarlo en el "marketplace" de Eclipse o abrirlo a la comunidad para que mejore constantemente.
   f. Añadir sistemas de visualización gráfica que sean dinámicos.

ii) A nivel de metodología:
   a. Preparar la aplicación y/o clases para realizar inferencias de nuevos documentos en base a los entrenamientos que ya se tienen realizados.
   b. Establecer sistemas para depurar los tokens al máximo. Añadir nuevos ajustes en el proceso de "stopwords" o palabras vacías, lematización y filtrado.
   c. Establecer y probar ajustes de parámetros configurables de LDA en MALLET para afinar más en el proceso y mejorar el resultado de las consultas.
   d. Establecer nuevos conceptos de uso como puedan ser: filtrados colaborativos, clasificación de documentos, perplejidad, polisemia, intercambiabilidad y estabilidad de topics. Binkley et al. (18), Bley y Lafferty (14) y Steyvers y Griffiths (51) hablan extensamente sobre cómo conseguir dichos conceptos.

9. BIBLIOGRAFÍA.

(Por orden de cita)

1. **Upadhyaya, Mamatha.** *Impact of big data on analytics.* Big Data, Capgemini. s.l. : Capgemini, 11 julio 2014. págs. 1-27, Presentación diapositivas en INSlideShare. https://es.slideshare.net/capgemini/impact-of-big-data-on-analytics.

2. **Loukides, Mike.** *What Is Data Science?* s.l. : CreateSpace Independent Publishing Platform , 15 de agosto 2014. pág. 156. ISBN-13: 978-1500839086.

3. *Probabilistic topics models.* **Blei, David M.** 4 de abril 2012, Princeton, N.J. : ACM, April 2012, Communications of the ACM, Vol. 55, pp. 77-84. doi:10.1145/2133806.2133826.

4. **Navarro Colorado, Borja.** Tema 6: Text Mining con Topic Modeling. [En línea] 2014. [Citado el: 24 de 10 de 2018.]

5. **Matuszeck, Paula.** CSC 9010: Aplicaciones de minería de datos. *Department of Computing Sciences.* [En línea] Villanova University, 2003. [Citado el: 7 de junio de 2019.] www.csc.villanova.edu/~matuszek/fall2003/index.html.

6. **Gardner, Andrew.** *Big Data and the Art of Data Science.* www.momentics.com. s.l. : Momentics, 21 marzo 2014. págs. 1-35, Presentación diapositivas en INSlideShare. https://www.slideshare.net/andrewgardner5811/big-data-and-the-art-of-data-science.

7. **Neri Cano, Héctor E.** *De qué hablamos cuando hablamos de Data Science.* DataLab Community. s.l. : DataLab Community, 7 agosto 2016. págs. 1-42, Presentación diapositivas en INSlideShare. https://es.slideshare.net/datalabmx/de-qu-hablamos-cuando-hablamos-de-data-science?qid=11c12af3-5e40-47f4-9ee4-d24af55545c7&v=&b=&from_search=8.

8. **Rivera, José L.** Introducción a Data Science. [En línea] 19 de agosto de 2015. [Citado el: 08 de febrero de 2020.] https://es.slideshare.net/SpanishPASSVC/introduccion-a-data-science?qid=11c12af3-5e40-47f4-9ee4-d24af55545c7&v=&b=&from_search=11.

9. **Bhatia, AshishSingh y Kaluza, Bostjan.** *Machine Learning in Java Second Edition. Helpful techniques to design, build, and deploy powerful machine learning applications in Java.* November 2018. Birmingham : Packt Publishing Ltd., 2018. pág. 290. ISBN 978-1-78847-439-9.

10. **Díaz Benito, Marta.** *Topic Model aplicado a letras de canciones.* Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, Universidad Politécnica de Madrid. 2017.

11. *Minería textual.* **Eito Brun, Ricardo y Senso, José A.** 1, s.l. : El profesional de la información, 2004, Vol. 13, págs. 11-27.

12. **López, Ander Carreño.** Detección de sucesos raros con Machine Learning. [ed.] U.P.M. *Trabajo Fin de Máster. Máster en Inteligencia Artificial.* Madrid, Madrid, España : Escuela Técnica Superior de Ingenieros Informáticos - UPM, 30 de junio de 2017. pág. 95.

13. **Blei, David M.;.** Topic Modeling and Digital Humanities. [En línea] Nº 1 de Vol. 2 de Winter 2012. [Citado el: 06 de septiembre de 2019.] http://journalofdigitalhumanities.org/2-1/topic-modeling-and-digital-humanities-by-david-m-blei/.

14. **Blei, David M. y Lafferty, John D.** *Topic Models.* Computer Science Department, Columbia University at the city of New York. New York : s.n., 2009. pág. 24. URL = http://www.cs.columbia.edu/~blei/papers/BleiLafferty2009.pdf.

15. **Santhanam, Deepak.** Latent Dirichlet Allocation. [En línea] 03 de marzo de 2010. [Citado el: 30 de enero de 2019.] http://cs.brown.edu/courses/csci2950-p/spring2010/lectures/2010-03-03_santhanam.pdf.

16. **Silvestre Gómez, María.** *Implementación de Asignación Jerárquica Latente de Dirichlet para Modelado de Temas.* Departamento Teoría de la Señal y Comunicaciones, Escuela Técnica Superior de Ingeniería. Universidad de Sevilla. Sevilla : s.n., 2018. pág. 103, Trabajo Fin de Máster.

17. *What is wrong with topic modeling? And how to fix it using search-based software engineering.* **Agrawal, Amritanshu, Menzies, Tim y Fu, Wei.** s.l. : Elsevier, 21 de junio de 2018, Information and Software Technology, Vol. 98, págs. 74-88. https://doi.org/10.1016/j.infsof.2018.02.005. ISSN=0950-5849.

18. *Understanding LDA in Source Code Analysis.* **Binkley, David, y otros.** [ed.] ACM. Hyderabad, India : s.n., 2014. ICPC.

19. **Linstead, Erik, y otros.** *Software Analysis with Unsupervised Topic Models.* University of California, Irvine y Chapman University, Orange, CA. págs. 1-4.

20. *Aplicaciones de la técnica de topic model en repositorios de software.* **López-Nozal, Carlos, y otros.** [ed.] José Riquelme, Alicia Troncoso y Salvador García. Granada : Asociación Española para la Inteligencia Artificial (AEPIA), 2018. IX Simposio de Teoría y Aplicaciones de la Minería de Datos. págs. 867-872. https://sci2s.ugr.es/caepia18/proceedings/docs/CAEPIA2018_TAMIDA.pdf.

21. **Saeidi, Amir M., y otros.** *ITMViz: Interactive Topic Modeling for Source Code Analysis.* Departament of Information and Computing Sciences, Utrech University. Artículo.

22. *JSEA: A Program Comprehension Tool Adopting LDA-based Topic Modelin.* **Wang, Tianxia y Liu, Yan.** [ed.] Dr Kohei Arai. 3, s.l. : The Science and Information (SAI) Organization Limited, 2017, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 8, págs. 433-437.

23. *Software Traceability with Topic Modeling.* **Asunción, Hazeline U., Asunción, Arthur U. y Taylor, Richard N.** [ed.] Jeff Kramer y Judith Bishop. Ciudad del Cabo, Sudáfrica : Association for Computing Machinery,New York (Estados Unidos), 2-8 Mayo 2010. ACM/IEEE 32nd International Conference on Software. ISBN: 978-1-60558-719-6.

24. *Orion: A Software Project Search Engine with Integrated Diverse Software Artifacts.* **Bissyandé, Tegawandé F., y otros.** Singapur : IEEE Computer Society Digital Library, Jul 17, 2013 - Jul 19, 2013. 2013 International Conference on Engineering of Complex Computer Systems. Vol. 1, págs. 242-245. DOI Bookmark:10.1109/ICECCS.2013.42.

25. *TopicNets: Visual Analysis of Large Text Corpora with Topic Modeling.* **Gretarsson, Brynjar, y otros.** ACM Journal, págs. 1-26.

26. **Lapeña, Raúl, Pérez, Francisca y Cetina, Carlos.** *On the Influence of Models-to-Natural-Language Transformation in Traceability Link Recovery among Requirements and Conceptual Models.* Valencia : Proceedings of the ER Forum 2017 and the ER 2017 Demo Track, Noviembre 2017. pág. 14, Presentación.

27. *Automatic query reformulations for feature location in a model-based family of software products.* **Pérez, Francisca; Font, Jaime; Arcega, Lorena; Cetina, Carlos;.** [ed.] P.P. Chen. s.l. : Elsevier B.V., Julio de 2018, Data & Knowledge Engineering, Vol. 116, págs. 159-176. https://doi.org/10.1016/j.datak.2018.06.001.

28. *Automatically classifying source code using tree-based approaches.* **Viet Phan, Anh, y otros.** [ed.] P.P. Chen. s.l. : Elsevier B.V., Marzo de 2018, Data & Knowledge Engineering, Vol. 114, págs. 12-25. https://doi.org/10.1016/j.datak.2017.07.003.

29. **Alores.** Entrevista a Javier Garzás, el mayor referente sobre la agilidad de habla hispana. [En línea] Velneo, 2 de abril de 2019. [Citado el: 11 de abril de 2020.] https://velneo.es/entrevista-a-javier-garzas-el-mayor-referente-sobre-la-agilidad-de-habla-hispana/.

30. **Vega López, Miguel.** Proceso Software Personal. *Departamento de Lenguajes y Sistemas Informáticos.* [En línea] Universidad de Granada. [Citado el: 10 de abril de 2020.] http://lsi.ugr.es/~mvega/docis/psp.html.

31. **Humphrey, Watts S.** The Personal Software Process (PSP). [En línea] Software Engineering Institute. Carnegie Mellon University, Noviembre de 2000. [Citado el: 11 de abril de 2020.] https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283.

32. **Wikipedia.** Personal Software Process. [En línea] Wikipedia, 14 de noviembre de 2019. [Citado el: 11 de abril de 2020.]

33. **Palacio, Marta.** Scrum Master. Temario Troncal 1. Versión 3.02. *Scrum Manager.* [En línea] marzo de 2020. [Citado el: 10 de abril de 2020.] https://scrummanager.net/files/scrum_master.pdf.

34. **Menzinsky, Alexander, López, Gertrudis y Palacio, Juan.** Scrum Master. Scrum Manager: Temario Troncal I.Versión 2.61 Enero 2019. *Scrum Master.* [En línea] enero de 2019. [Citado el: 19 de noviembre de 2019.]

35. **Palacio, Juan.** Scrum Level. Versión 3.11. Abril 2020. *Scrum Manager.* [En línea] abril de 2020. [Citado el: 10 de abril de 2020.] https://scrumlevel.com/files/scrumlevel.pdf.

36. **Schwaber, Ken y Sutherland, Jeff.** La Guía de Scrum™. La Guía Definitiva de Scrum: Las Reglas del Juego. [En línea] 2017. [Citado el: 14 de noviembre de 2019.] https://www.scrum.org/resources/scrum-guide.

37. **Sutherland, Jeff.** *Scrum.* [ed.] Inc., 2014 Jeff Sutherland y Scrum. Kindle - Amazon. s.l. : Editorial Planeta S.A., 2015. ISBN: 978-84-08-13701-6 (epub).

38. **PMOInformatica;.** Plantilla del registro de interesados. [En línea] 1 de abril de 2015. [Citado el: 14 de noviembre de 2019.] http://www.pmoinformatica.com/2015/04/plantilla-registro-de-interesados.html.

39. **McCallum, Andrew Kachites.** MALLET: A Machine Learning for Language Toolkit. [En línea] UMASS AMHERST, 2002. [Citado el: 27 de marzo de 2020.] http://mallet.cs.umass.edu/.

40. *Studying software evolution using topic models.* **Thomas, Stephen W., y otros.** [ed.] A. De Lucia M.R. Mousavi. s.l. : Eselvier B.V., Agosto de 2012, Science of Computer Programming, págs. 1-23. ISSN: 0167-6423.

41. **Burton, Matt.** The Joy of Topic Modeling. *A bag of words by Matt Burton on the 21st of May 2013.* [En línea] 21 de mayo de 2013. [Citado el: 06 de septiembre de 2019.] http://mcburton.net/blog/joy-of-tm/.

42. **Chandía Sepúlveda, Bruno.** *Aplicación y evaluación LDA para asignación de tópicos en datos de Twitter.* Facultad de Ingeniería. Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso. Valparaiso : s.n., Marzo 2016.

43. **Calvo, José.** Topic Modeling: ¿qué, cómo, cuándo? [En línea] More Than Books, 01 de diciembre de 2016. [Citado el: 03 de diciembre de 2018.] http://www.morethanbooks.eu/topic-modeling-introduccion/.

44. **Raj Baral, Dilip.** Topic Modelling using LDA with MALLET. [En línea] 4 de junio de 2017. [Citado el: 12 de diciembre de 2019.] https://diliprajbaral.com/2017/06/04/topic-modelling-lda-mallet/.

45. *Latent Dirichlet Allocation.* **Blei, David M.; Ng, Andrew Y.; Jordan, Michael I.** [ed.] John Lafferty. enero de 2003, Journal of Machine Learning Research, Vol. 3, págs. 993-1022. http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf y http://www.jmlr.org/papers/v3/blei03a.html.

46. *Distributed Algorithms for Topics Models.* **Newman, y otros.** s.l. : Journal Machine Learning Research, 2009.

47. *Efficient Methods for Topic Model Inference on Streaming Document Collections.* **Yao, Mimno, David y McCallum, Andrew.** [ed.] Association for Computer Machinery (ACM). s.l. : K.D.D., 2009.

48. **Blewitt, Alex;.** *Eclipse Plug-in Development Beginner's Guide Second Edition.* Second published: July 2016. Birmingham : Packt Publishing Ltd., 2016. p. 458. ISBN: 978-1-78398-069-7.

49. **Salgado, Carlos.** Sketchs, mockups, wireframes y prototipos. [En línea] Univesitat Oberta de Catalunya, 15 de septiembre de 2015. [Citado el: 27 de mayo de 2020.] https://mosaic.uoc.edu/2015/09/15/proceso-de-desarrollo-de-un-proyecto-digital/.

50. **Instituto Nacional de Estadística (I.N.E.).** Encuesta Trimestral de Coste Laboral (ETCL).Coste laboral por hora efectiva por divisiones de la CNAE-09. *62 Programación, consultoría y otras actividades relacionadas con la informática.* [En línea] 17 de marzo de 2020. [Citado el: 16 de abril de 2020.] https://www.ine.es/jaxiT3/Datos.htm?t=6037#!tabs-tabla.

51. **Steyvers, Mark y Griffiths, Tom.** Probabilistic Topic Model. [aut. libro] Laurence Erlbaum. [ed.] In T. Landauer, y otros. *Latent Semantic Analysis: A Road to Meaning.*

52. **Explained, Eurostat Static.** Salarios y costes laborales. [En línea] Eurostat Static Explained. Perteneciente a Eurostat., 11 de 1 de 2019. [Citado el: 19 de 05 de 2019.] https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Wages_and_labour_costs/es. ISSN 2443-8219.

53. **Google.** Recorrido Yecla - USJ (Villanueva de Gállego). *Escuela de Arquitectura y Tecnología USJ.* [En línea] Google. [Citado el: 18 de 05 de 2019.] https://www.google.es/maps/dir/Yecla,+Yecla/Escuela+de+Arquitectura+y+Tecnolog%C3%ADa+USJ,+Autov.+A-23+Zaragoza-Huesca,+Km+299,+50830+Villanueva+de+G%C3%A1llego+-Zaragoza,+Zaragoza/@41.6201485,-0.9977918,11z/data=!4m19!4m18!1m10!1m1!1s0xd63fdbaa8669b7b:0x.

54. **Hammoe, Luciano.** *Detección de tópicos utilizando el modelo LDA.* Instituto Tecnológico de Buenos Aires - ITBA, Escuela de Ingeniería y Tecnología - Ingeniería y Gestión - Postgrado. Buenos Aires : s.n., Segundo cuatrimestre 2018. Trabajo final presentado para la obtención del título de Especialista en Ciencia de Datos.

55. **Instituto Nacional de Estadística (I.N.E.).** Encuesta Trimestral de Coste Laboral (ETCL).Coste laboral por hora efectiva por divisiones de la CNAE-09. *62 Programación, consultoría y otras actividades relacionadas con la informática.* [En línea] 19 de 03 de 2019. [Citado el: 13 de 05 de 2019.] http://www.ine.es/jaxiT3/Datos.htm?t=6037.

56. —. 03 Métodos y proyectos /Glosario de Conceptos / Costes laborales. [En línea] 2019. [Citado el: 19 de 05 de 2019.] http://www.ine.es/DEFIne/es/concepto.htm?c=274&op=30187&p=1&n=20.

10. ANEXOS.

10.1. ANEXO 1: propuesta de proyecto autorizado.

universidad SANJORGE GRUPO SANVALERO

Propuesta de Proyecto Final

Nombre alumno: Valerio García Palao

Titulación: Graduado en Ingeniería Informática

Curso académico: 2018-2019

1. TÍTULO DEL PROYECTO

Utilizando Topic Modeling para buscar términos relacionados

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

El proyecto consiste en buscar términos (topics) en una colección de documentos que están relacionados con un texto proporcionado como entrada. Lo términos se buscarán utilizando la técnica *Topic Modeling*.

3. OBJETIVOS DEL PROYECTO

Los objetivos del proyecto son:

i) Realizar el diseño detallado de los componentes y estructuras de datos para abordar el problema planteado.
ii) Implementar una aplicación en Java que utilice la técnica *Topic Modeling*.

4. METODOLOGÍA

La metodología más adecuada se planteará por el alumno y se establecerá en las primeras fases del proyecto.

5. PLANIFICACIÓN DE TAREAS

Las tareas quedan predefinidas de manera global en los objetivos. Serán fijadas de forma concreta durante el desarrollo del proyecto

6. OBSERVACIONES ADICIONALES

FI-030 - 1 - Rev.0

Figura 38: propuesta de proyecto presentada.

10.2. ANEXO 2: contenido de CD.

La estructura del CD suministrado es la presentada en el siguiente esquema:

1. *MALLET*: instalación de MALLET lista para copiar al directorio correspondiente. Incluye pruebas iniciales, listas de stopwords empleadas y corpus creado.
2. *Documentación*.

   Toda la documentación se sirve en fichero PDF.

   1. Memoria del PFG: esta misma memoria.
   2. Manual básico de instalación y usuario.
   3. Pruebas realizadas: informe de pruebas iniciales realizadas con MALLET.
   4. Actas de reuniones de revisión de sprint y planificación del siguiente sprint.
   5. Tabla resumen de objetivos, requerimientos, requisitos y funcionalidades por sprint.
   6. Listado de stopwords.
3. *Código fuente proyecto Eclipse*.
   1. Plug-in: "usj.svit.topicmodeling.ui".

La Figura 39 presenta la estructura de directorios y contenidos del soporte físico anexo a la memoria impresa.

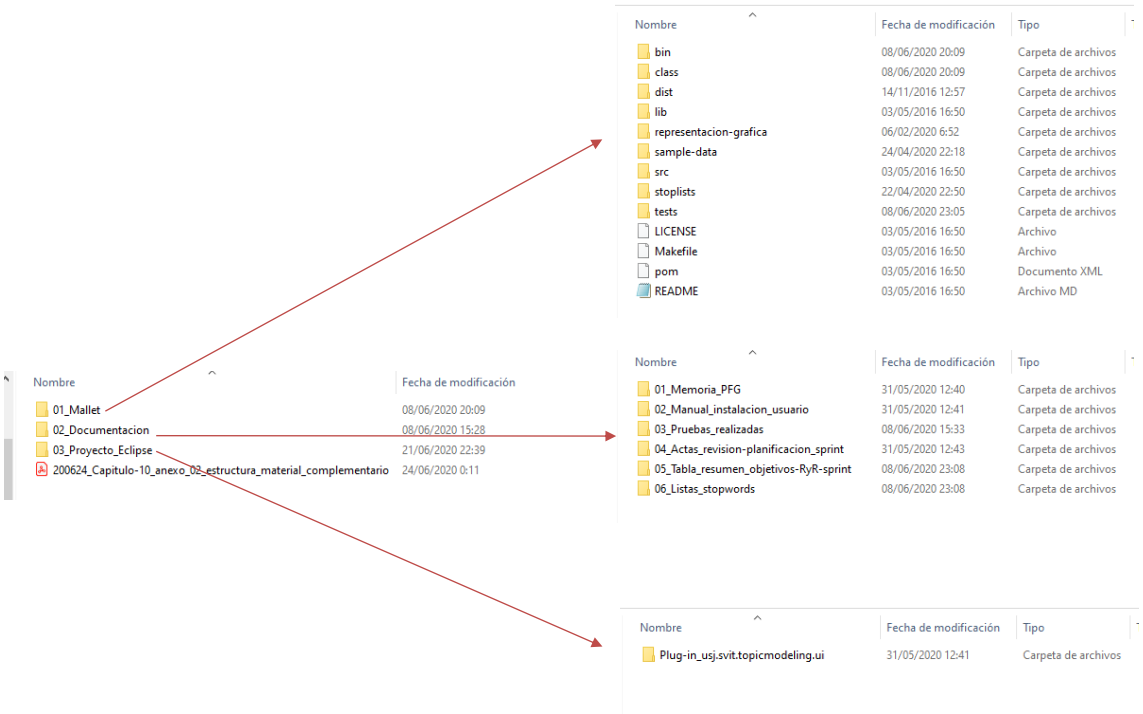


Figura 39: Esquema de directorios del soporte físico de material complementario.