Aplicaciones de la técnica de *topic model* en repositorios software

Carlos López-Nozal
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
Burgos, España
clopezno@ubu.es

César Ignacio García-Osorio
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
Burgos, España
cgosorio@ubu.es

Álvar Arnaiz-González
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
Burgos, España
alvarag@ubu.es

Mario Juez-Gil
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
Burgos, España
mariojg@ubu.es

***Resumen*—Los equipos de desarrollo software usan repositorios de proyectos, accesibles mediante forjas como Github, donde aplican un proceso iterativo e incremental. En el proceso se incluyen actividades como: gestión de tareas, gestión de versiones, codificación, pruebas, documentación, revisiones de calidad y despliegue de aplicaciones. Durante estas actividades se generan múltiples productos a través de complejas interacciones entre sus participantes. La caracterización tanto de productos como de interacciones basadas en texto es un primer paso para comprender y hacer más eficiente la gestión del proyecto. La técnica de aprendizaje textual denominada modelo de temas (*topic model*), mejora la comprensión de grandes cantidades de datos textuales agrupando los documentos en temas. Con el objetivo de poder mejorar las actividades de un proceso de desarrollo software, en este trabajo se presenta una revisión bibliográfica de cómo se está aplicando la técnica de aprendizaje automático y cuáles son los resultados de su aplicación en el contexto del proceso de desarrollo software.**

***Index Terms*—Model Topic, Software Repository, Text Mining, Software Development, Machine Learning**

I. Introducción

En la última década han surgido forjas de proyectos software de fácil acceso tanto para proyectos empresariales como para proyectos *OpenSource* (SourceForge Github, GitLab, Bitbucket). Estas forjas suelen integrar múltiples sistemas para dar soporte a los flujos de trabajo y registrar las interacciones entre los miembros del equipo.

Debido al interés que ha surgido en este campo, la comunidad científica creó en 2004 la Conferencia *Internacional Mining Software Repositories* (MSR http://www.msrconf.org/) como un punto de encuentro para analizar los datos disponibles en los repositorios software para descubrir información interesante y procesable sobre sistemas de software y proyectos.

Una de las líneas de trabajo en el aprendizaje automático es el tratamiento de grandes cantidades de documentos basados en texto para poder extraer relaciones entre ellos. El modelado de temas (*topic model*) es una técnica de agrupamiento de documentos de texto (*clustering*). Cada tema se define siguiendo una distribución de probabilidades de un conjunto de palabras que ocurren con más frecuencia en los documentos de ese tema.

Muchas actividades de los repositorios software llevan implícita una comunicación textual entre los miembros del equipo. Por ejemplo, el análisis de la información textual de las revisiones utilizando la técnica de modelado de temas puede mejorar la comprensión del proceso de calidad.

El objetivo de este trabajo es estudiar en la literatura científica cómo se está aplicando la técnica de modelado de temas, en los conjuntos de datos extraídos desde repositorios software.

El resto del artículo se estructura de la siguiente forma. En la Sec. II, se definen los conceptos teóricos necesarios para la comprensión del artículo. En la Sec. III, se detalla el proceso utilizado para seleccionar la bibliografía a revisar. Posteriormente, en la Sec. IV se resumen los resultados de la revisión bibliográfica. Finalmente, en la Sec. V, se presentan las conclusiones obtenidas del presente trabajo.

II. Definición de conceptos teóricos

La sección se ha organizado en dos subsecciones, una para los conceptos relacionados con la ingeniería de software y otra para los conceptos relacionados con ciencias de la computación y procesamiento de lenguaje natural.

II-A. Sistemas utilizados en el desarrollo software

Uno de los resultados del proceso interno de un equipo de desarrollo de software, es crear un conjunto de ficheros de texto que contienen un código que se puede desplegar y ejecutar en algún dispositivo electrónico. El código puede estar escrito en múltiples lenguajes de programación. Para gestionar todo este proceso de creación se usan varios sistemas

de gestión integrados en lo que se denomina repositorio del proyecto. A continuación se da una definición de repositorio de software y se describen alguno de sus sistemas gestión.

- Repositorio software. Los repositorios software son espacios virtuales donde los equipos de desarrollo generan los artefactos colaborativos procedentes de las actividades de un proceso de desarrollo. Los repositorios se alojan en lo que se denominan forjas de proyectos software (*GitHub, GitLab, SourceForge, Bitbucket*) En los repositorios además de guardar los artefactos, versión final y versiones previas, se almacena la interacción de los miembros del equipo justificando el cambio de versión. Dependiendo del artefacto generado se utilizan distintos sistemas: foros de comunicación, sistemas de control de versiones, sistemas de gestión de tareas.
- Sistema de control de versiones. Los *commits* son una de las entidades clave de estos sistemas. Cada cambio de versión de un fichero, o conjunto de ficheros, generado en el repositorio se conoce como *commit*. La evolución del software puede representarse como una secuencia de *commits*. Cada commit es creado por un miembro del equipo de desarrollo para registrar un cambio en el software. Además de los detalles de los cambios efectuados en los ficheros, un elemento importante en los *commits* es la información textual que incluye el desarrollador para describir la naturaleza del cambio (mensaje asociado al *commit*). Los repositorios están compuestos por ramas de desarrollo y estas contienen *commits*. Las ramas pueden estar en repositorios centrales, distribuidos en múltiples repositorios locales o remotos. La rama de desarrollo principal es la que mantiene los ficheros actualizados para un versión del software que está en explotación.
- Sistema de gestión de tareas. Cualquier actividad que se va a realizar en un repositorio debería ser notificada con una tarea (el término tarea también es referido como *issue* o *ticket*). De manera general, las actividades en un proceso de desarrollo de software se pueden categorizar en: desarrollo (implementación de nueva funcionalidad), documentación, mantenimiento (pruebas, reestructuración de código, corrección de errores), explotación (despliegue de la aplicación) y revisiones de calidad. Los elementos fundamentales de una tarea son: el texto descriptivo de la tarea, un estado que indica si está abierta o cerrada, opcionalmente puede tener etiquetas que ayuden a su clasificación. Además cada tarea tiene asociado un responsable y también tiene asociado el conjunto de comentarios textuales de los distintos miembros del equipo, desde su apertura hasta su cierre. Cada tarea tiene un identificador que sirve como componente integrador con el resto de sistemas. Por ejemplo el texto descriptivo de un *commit* puede incluir el identificador de la tarea para recoger una trazabilidad de la descripción de los cambios ocasionados por la tarea. Las peticiones de integración (*pull request*) son un tipo de tareas especiales que se crean automáticamente en el sistema de gestión de tareas cuando un miembro del equipo solicita una integración de sus cambios en la rama de desarrollo principal. Las integraciones pueden ser aceptadas o rechazadas después de un proceso/diálogo de discusión y revisión entre los miembros del equipo.

II-B. Agrupamiento y procesamiento de documentos

- Modelado de temas o *topic modeling*, es una técnica avanzada de recuperación de información que automáticamente encuentra los temas generales de en un conjunto de documentos de texto, llamado corpus, sin la necesidad de etiquetas, datos de entrenamiento o taxonomías predefinidas. El modelado de temas solo usa la frecuencias de las palabras y la co-ocurrencia de frecuencias en los documentos para construir un modelo de palabras relacionadas. Utilizando este enfoque simple, el modelado de temas se ha utilizado con éxito en múltiples dominios para organizar y analizar automáticamente millones de documentos no estructurados.
  En la actualidad existen varios algoritmos que sirven para implementar esta técnica, siendo los más referenciados LDA (Latent Dirichlet Allocation) [4], LSI (Latent Semantic Indexing) y HDP (Hierarchical Dirichlet Process). A continuación formalizamos matemáticamente este concepto. Dado un corpus de documentos $D = \{d_1, ..., d_n\}$ donde cada documento $d_i, i = 1, ..., n$ es una secuencia de m palabras denotadas por $d_i = \{w_1, ..., w_m\}$, $w_j \in W, j = 1, ..., m$. W es el vocabulario del conjunto de documentos. Cada documento d_i se puede modelar como una distribución multinomial θ^{d_i} sobre t temas, y cada tema z_k, $k = 1, ..., t$ se modela como una distribución multinomial ϕ^k sobre el conjunto de palabras W.
- Caracterización de documentos de texto. La técnica de modelado de temas necesita caracterizar cada documento $\{d\}$ con conjunto de valores de entrada $\{x_1, ..., x_n\}$. La bolsa de palabras (*Bag of Words*) es la aproximación más utilizada para representar documentos de texto. En esta representación cada palabra del conjunto de documentos es considerada como una característica x_j. Existen varios tipos de transformaciones:
  - Boolean: es la más simple de las transformaciones. Si una palabra w_j está presente en el documento, la característica x_j toma el valor de 1 y 0 en caso contrario.
  - Raw TF (*Term Frequency*): en esta transformación, la característica x_j toma como valor el número de ocurrencias de la palabra, w_j, en el documento d_i y se referencia como $f_{i,j}$. Aunque parece una representación más sofisticada, tiene la desventaja que palabras con poco significado a menudo tienen valores de frecuencias muy altos.
  - Escala logarítmica TF: esta versión de TF se usa para suavizar la desventaja mencionada de la representación Raw TF. Existen varias transformaciones logarítmicas, una de las más usada se calcula aplicando

Cuadro I
Ejemplo de tareas de procesamiento de lenguaje natural.

.token	.lemma	.tag	.es_alfabético	.palabra vacía
Updates	update	NOUN	True	False
link	link	VERB	True	False
text	text	NOUN	True	False
and	and	CONJ	True	True
URLs	url	NOUN	True	False
in	in	ADP	True	True
other	other	ADJ	True	True
docs	doc	NOUN	True	False

$\log(1 + f_{i,j})$ para cada palabra w_j en el documento d_i.

- IDF (*Inverse Document Frequency*): IDF asigna el valor $f_{i,j} \log(N/N_j)$, donde N es el número total de documentos del corpus, y N_j es el número de documentos que contienen la palabra. Aplicando esta trasformación, una palabra tiene más peso si es infrecuente en el corpus de documentos.
- TF-IDF: las transformaciones TF y IDF se multiplican. De esta forma se incrementa el valor de la característica cuando una palabra es frecuente en el documento pero infrecuente en el corpus de documentos.

- Procesamiento del lenguaje natural. La extracción de palabras de los documentos se obtiene aplicando una secuencia variable de tareas de procesamiento entre las que se pueden incluir las siguientes: *tokenización*, eliminación de palabras sin significado, selección de términos gramaticales (sustantivo, verbo, adjetivo, adverbio), identificación de secuencias de palabras que se utilizan juntas y lematización.
  La *tokenización* es la tarea encargada de dividir un texto en *tokens*. Cada token es una palabra contenida en el texto y está separado con delimitadores, normalmente son espacios en blanco y signos de puntuación. En el lenguaje existen palabras vacías que no aportan significado, como puede ser las preposiciones, artículos, etc. Se suele disponer de una lista de palabras vacías asociadas a cada idioma, generalmente denominada *stop words*. El proceso de identificación sintáctica de las palabras en una frase del documento se denomina etiquetado gramatical (conocido también por su nombre en inglés, *part-of-speech tagging, POS tagging o POST*). El lematizado es una forma de normalización de la palabra, reduce una palabra a su forma base, raíz o lema.
  En el Cuadro I se muestra un ejemplo ilustrativo de los resultados de aplicar las tareas de procesamiento de texto: tokenización, identificación de palabras vacías, etiquetado gramatical y lematización. El ejemplo es aplicado para procesar la siguiente entrada de texto "*Updates link text and URLs in other docs"*. Las columnas se corresponden con las salidas de las tareas del preproceso (token, lema, tag, stop, es alfabético) y las filas son tokens de la frase de entrada.

III. Proceso de selección de artículos

En esta sección se describe cuál ha sido el proceso de búsqueda y selección de artículos que sirvan para comprender cómo se está utilizando la técnica de modelado de temas en los repositorios software.

Se utiliza la base de datos Scopus para seleccionar los documentos de interés a revisar. Inicialmente la búsqueda con la palabra clave "GitHub", se hace en el título, resumen y palabras clave de los artículos. El número de artículos encontrados con este criterio de búsqueda fue de 3,697. En la siguiente iteración de búsqueda se incluye en la cadena de búsqueda "LDA" (*Latent Dirichlet Allocation*), por ser las siglas del algoritmo de modelado de temas más utilizado. Con este refinamiento del criterio de búsqueda, el número de documentos fue de 18. Posteriormente se refina la búsqueda incluyendo otros repositorios de proyectos y otros algoritmos de modelado de tópicos. Finalmente se obtienen 30 artículos siendo las principales áreas de conocimiento *Computer Science* (24) y *Mathematics* (5).

Como última etapa del proceso de la búsqueda sistemática, se realizó una lectura de los resúmenes y palabras clave para poder verificar la validez del artículo respecto a los criterios de búsqueda. En esa inspección se encontraron varios falsos positivos, por un lado debido a la aparición de la palabra reservada Github como url utilizada para distribuir una determinada implementación de software en un repositorio. Por otro lado, por el uso del término "lda" en la sentencia "*Science and Technology Publications, Lda*". Como resultado se obtuvieron ocho artículos, dos de revistas y seis de actas de congresos internacionales.

Además de este proceso de búsqueda sistemática, se realizó una búsqueda no sistemática utilizando múltiples criterios con términos relacionados: *machine learning*, *text mining*, *text clustering*, *mining software repositories*, ... Como resultado de este proceso se incluyeron tres artículos más de revistas.

El Cuadro II contiene una descripción de las referencias bibliográficas de estudio ordenadas por método de búsqueda (sistemático vs. no sistemático) y después por año de publicación.

IV. Revisión bibliográfica

Para facilitar la comprensión de esta revisión, en las siguientes subsecciones caracterizamos todos los trabajos seleccionados desde tres perspectivas: descripción del conjunto de datos que utilizan, aplicación del modelo de temas, técnicas de procesamiento del lenguaje natural.

IV-A. Descripción de los conjuntos de datos

En el Cuadro III se muestran los resultados de la caracterización para los que se han considerado las siguientes cinco características:

- CD1 Tipo de entidades, es un medida nominal que identifica los elementos de análisis dentro del repositorio software. Puede tomar los siguientes valores: *pull request, commit, source code, fichero Readme, issue, post*. En la Subsec. II-A se han explicado *pull request,*

Cuadro II
Artículos seleccionados para la revisión

Título	Tipo	Año	Referencia
What are developers talking about? An analysis of topics and trends in Stack Overflow	Revista	2014	[3]
Open source is a continual bugfixing by a few	Actas congreso	2014	[5]
An insight into the pull requests of GitHub	Actas congreso	2014	[12]
Mining source code topics through topic model and words embedding	Actas congreso	2016	[17]
Topic-Based Integrator Matching for Pull Request	Actas congreso	2017	[8]
Cataloging GitHub repositories	Actas congreso	2017	[13]
Developer Identity Linkage and Behavior Mining Across GitHub and StackOverflow	Revista	2017	[15]
Mining developer behavior across git hub and stack overflow	Actas congreso	2017	[16]
Mining software repositories for defect categorization	Revista	2015	[7]
MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks	Revista	2015	[14]
Understanding Review Expertise of Developers: A Reviewer Recommendation Approach Based on Latent Dirichlet Allocation	Revista	2018	[6]

Cuadro III
Resumen de las características de los conjuntos de datos experimentales de la bibliografía.

BIB	CD1	CD2	CD3	CD4	CD5
[3]	post	3 447 987	7 meses actividad	SI	StackOverFlow
[5]	commits, issues	NO	43	NO	2014 MSR Challenge
[12]	pull request	9 421	78	NO	2014 MSR Challenge
[17]	source code	NO	100	SI	2013 MSR Challenge
[8]	pull request	4 364	3	NO	Github
[13]	fichero Readme	10 000	10 000	SI	Github
[16]	issue y post	16 000	No especificado	SI	Github StackOverFlow
[7]	Issue-bug	2 500	4	NO	OpenSource
[14]	post, commit, bugs	NO	3	NO	OpenSource
[6]	pull request	1 345	5	NO	Github

commit, source code e issue. Por *post* se entiende el mensaje y respuestas enviadas por los desarrolladores en un foro de tipo pregunta respuesta, como *StackOverFlow*. Los ficheros *Readme* de los repositorios de GitHub, son ficheros de texto que sirven para describir el contenido del repositorio, son una página de presentación del repositorio.

- CD2 Número de entidades o documentos de texto en la validación empírica del algoritmo de modelado de temas.
- CD3 Número de repositorios incluidos en el diseño experimental.
- CD4 Uso de entidades de múltiples repositorios es una medida booleana. En el caso de ser cierta, indica que en el diseño experimental utiliza las entidades de múltiples repositorios mezcladas entre sí. En [3] toma el valor de cierto, porque extrae los temas asociados a los post de la plataforma StackOverFlow durante un periodo de siete meses de actividad. En [17] se agrupan por temas ficheros fuentes de múltiples repositorios y en [13] se analizan los ficheros *Readme* de múltiples repositorios para extraer sus temas automáticamente.
- CD5 Disponibilidad de acceso a los conjuntos de datos utilizados en el validación empírica es una medida nominal. Github y StackoverFlow indican que en los trabajos se utilizada un API (*Application Program Interface*) pública para acceder a datos de estos repositorios. *MSR Challenge* (*Mining Software Repositories*) son los conjuntos de datos utilizados para una sesión de desafío del conferencia internacional MSR. *OpenSource* se ha utilizado para categorizar los trabajos con información de repositorios de tipo *OpenSource*.

IV-B. Aplicación del modelo de tópicos

En la Tabla IV se muestran los resultados de la caracterización para los que se han considerado las siguientes seis características:

- AMT1 Algoritmo es una medida nominal con el nombre del algoritmo de modelado de temas. En la bibliografía existe varias implementaciones distintas del algoritmo LDA, se han identificado con un número, 1 para la implementación conocida como MALLET (http://mallet.cs.umass.edu/) y 2 para la implementación conocida como JGIBBLDA (http://jgibblda.sourceforge.net/). LDA-GA indica una combinación del uso de LDA con algoritmos genéticos (*Genetic Algorithms*) para la optimización de sus parámetros en un conjunto de datos concreto. EmbTE *Embedded Topic Extraction*, se corresponde con la solución particular presentada [17], al igual que SDCL *Software Defect CLustering* es la propuesta de solución en [7].
- AMT2 Iteraciones es un parámetro del algoritmo que sirve para determinar la condición de parada del algoritmo y convergencia de los resultados. Es un parámetro opcional.
- AMT3 Número de tópicos/temas es un parámetro necesario por el algoritmo LDA.
- AMT4 Etiquetado manual es una medida booleana. En el caso de ser cierta indica que los temas, probabilidades de conjuntos de palabras, obtenidos como salida del algoritmo son verificados manualmente por una persona. El objetivo de la verificación es mejorar la comprensión humana asignando una única palabra clave al conjunto palabras obtenidas con el algoritmo. Por ejemplo en el etiquetado manual de [3] se asigna "*SQL*" como palabra clave del tema cuyas cuatro palabras más probables son "*quer*", "*table*", "*sql*" y "*row*". El proceso de etiquetado manual, implica determinar el número de palabras con las que establecer la palabra clave (por ejemplo en [12] usan 4). Además, hay que tomar decisiones sobre cuáles de los temas obtenidos como salida del algoritmo son coherentes. En [13] durante las inspecciones : *i*) asignan una etiqueta a un tema, *ii*) eliminan un tema por no ser coherente y *iii*) fusionan varios temas en uno.
- AMT5 Validación es una medida booleana que indica si en el diseño experimental se incluye alguna calibración

Cuadro IV
Resumen de las características de la aplicación del algoritmo de modelado de la bibliografía.

BIB	AMT1	AMT2	AMT3	AMT4	AMT5	AMT6
[3]	LDA1	500	40	SI	NO	NO
[5]	LDA1	1000	50	NO	NO	NO
[12]	LDA2	3000	100	SI	NO	NO
[17]	LDA,EmbTE		NO	NO	SI	IDF -TF y TF
[8]	LDA2	1000	15	NO		NO
[13]	LDAGA	500	49	SI	SI	IDF-TF
[16]	LDA	NO	NO	NO	NO	NO
[7]	SDCL			NO		IDF-TF
[14]	LDAGA	NO	NO	NO	SI	NO
[6]	LDA1	NO	20	NO	NO	NO

Cuadro V
Resumen de las características de la aplicación del algoritmo de modelado de la bibliografía.

BIB	TP1	TP2	TP3	TP4	TP5	TP6
[3]	ALTO	SI	SI	NO	NO	2-grams
[5]	MEDIO	SI	NO	SI	NO	NO
[12]	BAJO	SI	SI	NO	NO	NO
[17]	MEDIO	SI	SI	SI	SI	NO
[8]	BAJO	SI	SI	NO	NO	NO
[13]	ALTO	SI	SI	NO	NO	NO
[16]	BAJO	SI1	SI	SI	NO	NO
[7]	BAJO	SI	SI	NO	NO	NO
[14]	MEDIO	SI	NO	NO	NO	NO
[6]	MEDIO	SI	SI	NO	NO	NO

de los parámetros del algoritmo. Por ejemplo en [13] se utilizan medidas de coherencia de los temas obtenidos con LDA para determinar el parámetro de número de tópicos óptimo. La utilización de algoritmos genéticos para la optimización de parámetros de LDA es una solución presentada en varios trabajos [13], [14].

- AMT6 Transformación del corpus es una medida nominal que indica la representación numérica del conjunto de documentos. Es uno de los parámetros de entradas obligatorios del algoritmo ya que no puede configurarse con un valor por defecto. Los posibles valores son los presentados en Subsc. II-B: TF, IDF-TF.

IV-C. Técnicas de procesamiento del lenguaje natural

En la Tabla V se muestran los resultados de la caracterización para los que se han considerado las siguientes seis características:

- TP1 Tokenización es una medida ordinal que puede tomar los valores BAJO, MEDIO y ALTO. BAJO hace referencia cuando el proceso de división de textos en palabras se basa únicamente en utilizar espacios en blanco y signos de puntuación como separadores. MEDIO cuando utiliza un sistema de tokenización especial relacionado con el tipo de documento. AVANZADO cuando realiza múltiples sistemas de tokenización especiales. Para aclarar el concepto de sistema de tokenización especial se presentan algunos casos concretos. En [13] se elimina texto de cabeceras de los ficheros *Readme* para quedarse sólo con aspectos funcionales, se eliminan texto relacionados con licencias, instalación... Cuando los documentos a tratar contienen código de programación, como en [6] y [17], se realiza un tipo de procesamiento especial denominado *camel case splitting* para dividir identificadores de las entidades de código que están compuestos por varias palabras, por ejemplo "*nameToIndex*", "*loanInterest*" o "*hasDupdName*". En [14] se eliminan identificadores de usuario dentro del cuerpo de los documentos, típicamente precedidos por el símbolo @.
- TP2 Palabras vacías es una medida booleana para indicar que se ha incluido esta tarea en el procesamiento del lenguaje. En [17] y en [6] se añaden las palabras que forman parte del lenguaje de programación (*if*, *implements*, *class*...).
- TP3 Lematización es una medida booleana para indicar que se ha incluido la tarea en el procesamiento del lenguaje. El tipo de algoritmo empleado mayoritariamente es el de Porter.
- TP4 Filtrado de palabras en el corpus es una medida booleana para indicar que se ha incluido esta tarea en el procesamiento del lenguaje. Los valores umbrales de las frecuencia de palabras en el corpus añaden poca información relevante para categorizar. En [5] se eliminan cadenas de palabras muy frecuentes en el texto de las *issues*: "*good job*", "*thank you*". En [17] elimina palabras muy frecuentes en los identificadores de entidades de programación como por ejemplo "*set*". En [15] se aplica una estrategia más general eliminando todas las palabras que tienen una frecuencia inferior a 20 en el conjunto de documentos.
- TP5 Etiquetado sintáctico es una medida booleana para indicar que se ha incluido esta tarea en el procesamiento del lenguaje. Además del etiquetado de lenguaje natural explicado en Subsec. II-B se considera etiquetado sintáctico la identificación de entidades de código: clase, método, parámetro, etc. que se aplica en [17] sobre los documentos de texto basados en código fuente.
- TP6 Identificación de secuencia de palabras que se usan juntas es una medida booleana para indicar que se ha incluido esta tarea en el procesamiento del lenguaje. En [3] se muestra el siguiente ejemplo de aplicación de está técnica "*compile time error*", tiene tres uni-grams "*compile*", "*time*", "*error*" y 2-grams ("*compile_time*", "*time_error*"). Para aplicar está técnica configura un parámetro de la implementación MALLET del algoritmo LDA, llamado *gram-size*.

V. Conclusiones

En la actualidad, empieza a existir un interés en aplicar la técnica de modelado de temas en las actividades de los sistemas de los repositorios software. Su objetivo es mejorar las tareas de mantenimiento del software caracterizando las entidades de interacción textual. Esta observación está basada en que la búsqueda sistemática de bibliografía ha localizado diversos artículos comprendidos entre las fechas 2014 y 2018.

En la revisión bibliográfica, la aplicación del modelado de temas ha sido motivada por diferentes causas. Una de ellas ha sido la aplicación de manera directa sobre alguna entidad textual de los repositorios, con el objetivo de mejorar la comprensión. Otra motivación es utilizar las salidas del algoritmo de modelado de temas como entrada de otras técnicas de aprendizaje automático.

Respecto a la caracterización de los conjuntos de datos experimentales del Cuadro III, se observa que el modelado de temas se aplica sobre múltiples entidades de los repositorios (CD2), y que el número de entidades varía mucho, desde 3,447,987 hasta 1,345. Los valores altos de CD2 se corresponden con trabajos donde se aplica el modelado de temas con entidades de múltiples repositorios (CD4). Esta correlación también ocurre con el número de repositorios (CD3, CD4). Algunos de los trabajos que realizan diseños experimentales aplicando LDA sobre un único repositorio, eliminan inicialmente algunos repositorios por falta de número de entidades y de contenido textual para experimentar. El acceso a los conjuntos de datos es abierto, pero necesitan eliminar ruido para mejorar la experimentación. Una carencia importante observada es que no hay ningún trabajo que experimente sobre datos de empresa que no procedan de proyectos *OpenSource*.

Respecto a la caracterización de cómo se aplica el algoritmo de modelado de temas del Cuadro IV, se observa que LDA es un algoritmo de referencia en el modelado de temas, pero la configuración de sus parámetros varía entre experimentos. Este es el caso del número de iteraciones (AMT2) y número de temas (AMT3). Además, otros parámetros rara vez se incluyen en el artículos, este es el caso de la transformación del corpus utilizada (AMT6). La transformación del corpus predominante en lo trabajos es IDF-TF. En el 30 % de los trabajos, la salida de LDA es supervisada por una persona para garantizar la coherencia de los temas obtenidos. Solo el 30 % de los trabajos validan empíricamente que los parámetros elegidos para la ejecución del algoritmo tiene un desempeño óptimo (AMT5). Una tendencia actual es validar estos parámetros utilizando algoritmos genéticos sobre diferentes ejecuciones del algoritmo con distintos parámetros. En este sentido en la bibliografía se han identificado dos nuevas fuentes bibliográficas [11] y [1] que detallan como calibrar los parámetros de LDA.

Respecto a la caracterización de técnicas del lenguaje natural recogidas en el Cuadro V, se observa que las técnicas clásicas de eliminación de palabras vacías (TP1) y lematización (TP3) con el algoritmo de Porter son especificadas en el 100 % de los trabajos. Otras técnicas documentadas que mejoran la calidad de la representación del documentos como: filtrado de términos por frecuencia en el corpus documentos (TP4), análisis de elementos sintácticos (TP5) e identificación de secuencia de palabras (TP6) son aplicadas en el 30 % de los artículos, 10 % y 10 % respectivamente.

Como conclusión final se considera que el modelado de temas aplicado a tareas de mantenimiento del software tiene una proyección prometedora, pero su aplicación necesita madurar. En este sentido, en los artículos revisados no se ha encontrado la documentación experimental necesaria que facilite el replicado del experimento, es decir, repositorios *OpenSource* con conjuntos de datos y los programas para comparar y mejorar los diseños experimentales.

Agradecimientos

Este trabajo ha sido financiado por el proyecto TIN2015-67534-P (MINECO/FEDER, UE) del Ministerio de Economía y Competitividad del Gobierno de España y por el proyecto BU085P17 (JCyL/FEDER, UE) de la Junta de Castilla y León ambos cofinanciados con los fondos FEDER de la Unión Europea.

Un especial agradecimiento a la Doctora Yulan He de la Aston University por acogernos en sus seminarios y enseñarnos técnicas sobre procesamiento del lenguaje natural y aprendizaje automático basado en texto.

Referencias

[1] Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, jun 2018.

[2] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. Software traceability with topic modeling. volume 1, pages 95–104, 2010.

[3] A. Barua, S.W. Thomas, and A.E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, 2014.

[4] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[5] M. Fejzer, M. Wojtyna, M. Burzańska, P. Wiśniewski, and K. Stencel. Open source is a continual bugfixing by a few. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8716:153–162, 2014.

[6] Jungil Kim and Eunjoo Lee. Understanding review expertise of developers: A reviewer recommendation approach based on latent dirichlet allocation. *Symmetry*, 10(4), 2018.

[7] S. Kumaresh and R. Baskaran. Mining software repositories for defect categorization. *Journal of Communications Software and Systems*, 11(1):31–36, 2015.

[8] Z. Liao, Y. Li, D. He, J. Wu, Y. Zhang, and X. Fan. Topic-based integrator matching for pull request. volume 2018-January, pages 1–6, 2018.

[9] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn. Bug localization using latent dirichlet allocation. *Information and Software Technology*, 52(9):972–990, 2010.

[10] G. Maskeri, S. Sarkar, and K. Heafield. Mining business topics in source code using latent dirichlet allocation. pages 113–120, 2008.

[11] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, may 2013.

[12] M.M. Rahman and C.K. Roy. An insight into the pull requests of github. pages 364–367, 2014.

[13] A. Sharma, F. Thung, P.S. Kochhar, A. Sulistya, and D. Lo. Cataloging github repositories. volume Part F128635, pages 314–319, 2017.

[14] X. Sun, B. Li, H. Leung, B. Li, and Y. Li. Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks. *Information and Software Technology*, 66:1–12, 2015.

[15] Y. Xiong, Z. Meng, B. Shen, and W. Yin. Developer identity linkage and behavior mining across github and stackoverflow. *International Journal of Software Engineering and Knowledge Engineering*, 27(9-10):1409–1425, 2017.

[16] Y. Xiong, Z. Meng, B. Shen, and W. Yin. Mining developer behavior across git hub and stack overflow. pages 578–583, 2017.

[17] W.E. Zhang, Q.Z. Sheng, E. Abebe, M. Ali Babar, and A. Zhou. Mining source code topics through topic model and words embedding. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10086 LNAI:664–676, 2016.