Recomendador de artículos científicos basado en metadatos de repositorios digitales

Article · September 2015

CITATIONS
0

READS
164

3 authors, including:

Ricard de la Vega
Consorci de Seveis Universitaris de Catalunya (CSUC), Barcelona, Spain
28 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Recomendador de artículos científicos basado en metadatos de repositorios digitales

Ruben Boada, Ricard de la Vega, Ángel Carreño

Proyecto Final de Postgrado

I Postgrado en Big Data Management and Analitycs
Universitat Politècnica de Catalunya (UPC-BarcelonaTech)

9 de septiembre de 2015

Technical advisor: Óscar Romero
Prototype advisor: Víctor Herrero
Business advisor: Pere Torrents

Índice

keywords: recommendations, content-filtering, research papers, big data, hadoop, hdfs, hbase, spark, couchbase, oai-pmh, digital library

1 Introducción

El uso de repositorios institucionales[1] es una tendencia en auge ya que representa una fuente de conocimiento fácilmente accesible, permitiendo una interactividad entre investigadores que deciden publicar su trabajo y conocimiento en abierto (Open Access[2]), y usuarios, muchos de ellos también investigadores, que encuentran en los repositorios unas fuentes de datos estructuradas.

La idea es generar conocimiento compartiendo y abriendo el acceso a la producción científica, facilitando la labor conjunta de investigadores e instituciones. Dentro de esta corriente de cooperación es donde se ubica el proyecto, buscando la interacción entre diferentes repositorios para facilitar al usuario el acceso a los datos, independientemente de en qué repositorio estén ubicados.

En ese sentido, una de las herramientas más usada actualmente, como son los recomendadores de contenido, no tienen casi presencia o su presencia se limita a recomendaciones sencillas y sin la cooperación inter-repositorios.

El documento se estructura en once apartados. Tras una breve introducción se analiza la necesidad del proyecto y se detallan los objetivos. Después se trata el estado del arte de este tipo de herramientas diferenciando nuestra aproximación y se describe el modelo de negocio. En el sexto apartado de describen los datos y en el séptimo la arquitectura del proyecto. Seguidamente de justifica la selección de las herramientas usadas, se realiza un análisis cuantitativo de la solución y se detallan los resultados obtenidos. En el último apartada se presentan las conclusiones y se describen las futuras líneas de trabajo.

2 Necesidad

La idea del proyecto surge de la necesidad de los investigadores de ahorrar tiempo en la búsqueda de información. Muchos de ellos se encuentran en un entorno altamente competitivo donde sólo encuentra el éxito quien consigue publicar antes un avance.

Este entorno competitivo a veces dificulta la recomendación entre colegas, de quién si no son de un círculo próximo, a veces se desconfía. Les gustaría que tecnología altamente eficaz que usan en su día a día, como las recomendaciones de libros de Amazon, también se pudiera aplicar a su ámbito profesional, por ejemplo, recomendando eficazmente publicaciones científicas.

En la figura 2 se muestra un mapa de empatía donde se detalla la necesidad de los investigadores descrita en este apartado.

[1] `https://en.wikipedia.org/wiki/Institutional_repository`
[2] `https://en.wikipedia.org/wiki/Open_access`

Figure 1: Empathy Map de los investigadors

Por otro lado, las instituciones que disponen de repositorios digitales, como podrían ser universidades o centros de investigación, tienen la necesidad de ofrecer servicios de valor añadido en estos productos para que sean del agrado de los investigadores. Para ellas es importante ofrecerles un buen servicio. A parte, cuanto mayor y más usado sea el repositorio, más puntos conseguirá la institución en algún *ranking* comparativo.

En la figura 2 se muestra un segundo mapa de empatía desde el punto de vista de las instituciones que disponen de un repositorio digital.

3 Objetivo

El objetivo del proyecto es **realizar un recomendador de artículos científicos**.

Concretamente, a partir de metadatos de artículos recolectados de repositorios digitales mundiales, se quiere realizar un proceso de recomendación basado en contenido a nivel de artículo. Para cada artículo se quiere calcular los N artículos más "similares", y ofrecer estos resultados via API para que se puedan integrar en las páginas de artículos de los propios repositorios digitales.

Algunos repositorios actuales disponen de recomendadores a nivel de artículo, però normalmente se basan en recomendar artículos del mismo autor o materia pero siempre del propio repositorio, pero no se ha encontrado un recomendador "global" que recomiende tanto de artículos internos como externos al propio repositorio basado en metadatos de "todos" los repositorios existentes.

3.1 Alcance del prototipo

Se considera que potencialmente todos los repositorios digitales mundiales (más de 3.000[3]) podrían ser la fuente de datos del recomendador. Sin embargo, se ha considerado que para el prototipo será suficiente con un par de repositorios ya que si se escoge alguno importante, como Pubmed Central[4], se dispondrá de un gran volumen de contenidos.

Un objetivo de la realización del prototipo es la obtención de métricas de rendimiento que permitan realizar estimaciones del sistema completo y analizar su escalabilidad y viabilidad

4 Trabajos relacionados

Los sistemas de recomendación [1, 2] son técnicas y herramientas *software* que proporcionan sugerencias de ítems para usuarios. Las sugerencias están relacionadas a diversos procesos de decisión, como qué libro comprar o qué música escuchar. El término ítem generalmente se usa para denotar el concepto a sugerir. Estos sistemas acostumbran a producir una lista de recomendaciones a través de dos aproximaciones: basados en filtrado colaborativo (*collaborative filtering*) o por contenido (*content filtering*).

[3] `http://www.opendoar.org/countrylist.php`

[4] `http://www.ncbi.nlm.nih.gov/pmc`

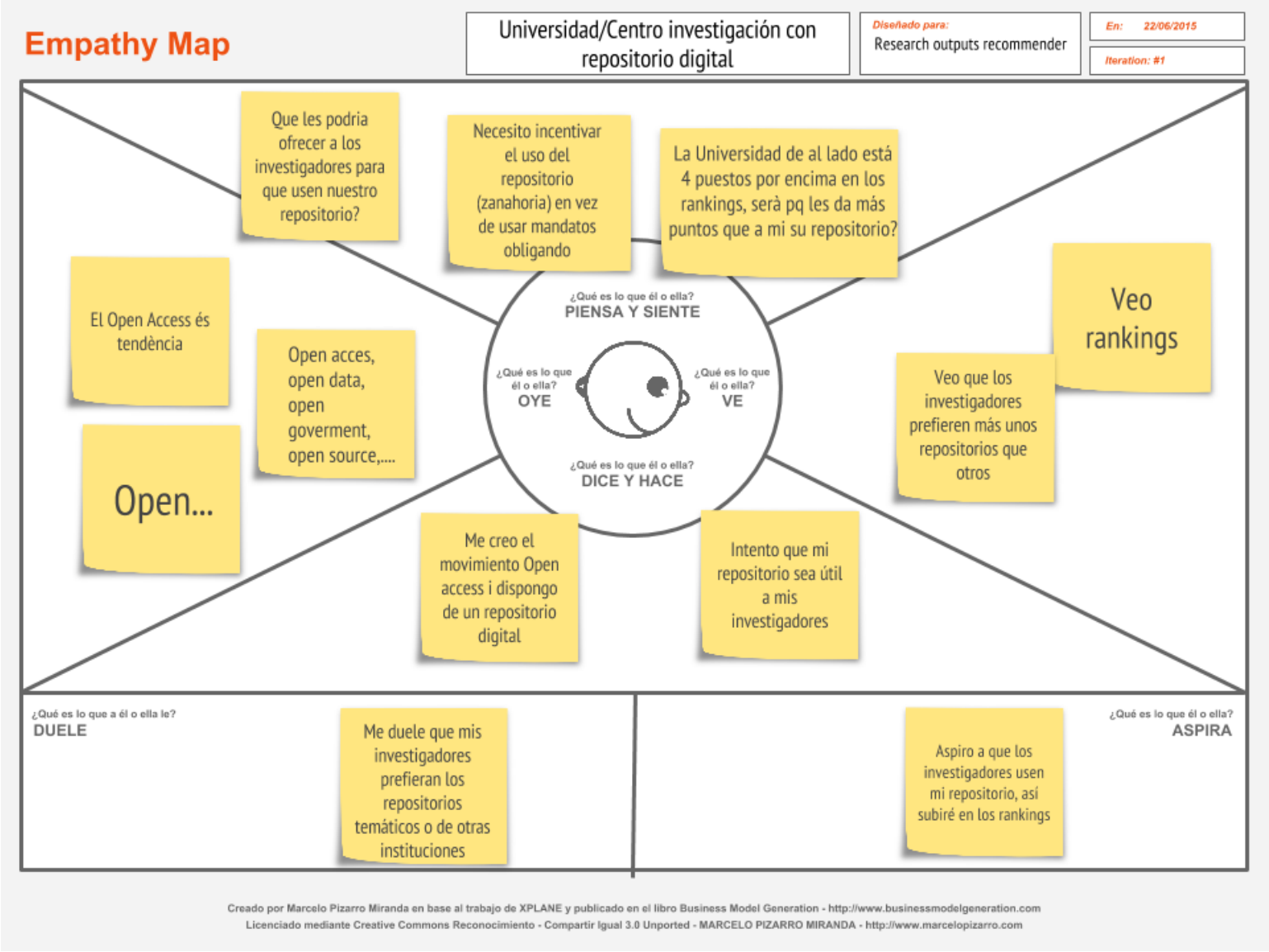


Figure 2: Empathy Map de las instituciones que disponen de repositorio digital

El modelo colaborativo construye un modelo basado en el histórico del usuario, por ejemplo, con ítems comprados anteriormente, o puntuaciones (*ratings*) con las que se ha calificado ítems con anterioridad, así como decisiones similares de otros usuarios. Este modelo se usa para predecir ítems (o puntuaciones sobre ítems) en los que el usuario puede estar interesado. Existen dos tipos de filtrado colaborativo, los *memory-based*, que trabajan con todos los datos, y los *model-based*, que construyen el modelo basado en estimaciones.

En cambio, el modelo de filtrado por contenido utiliza una serie de características discretas del propio item para recomendar otros ítems con propiedades similares. Las dos aproximaciones suelen combinarse en lo que se ha llamado filtrado híbrido (*hibrid filtering*).

Los sistemas de recomendación se han concentrado tradicionalmente en el sector del comercio electrónico (Amazon [3]...), amigos (Facebook [7]...), colegas (LinkedId [8]...), citas [5] y en el audiovisual, recomendando películas (Netflix [4]...), libros (Amazon [3]...), canciones (Spotify [9]...), etc. Pero en los últimos años se ha extendido a la comunidad científica, por ejemplo, para recomendar artículos científicos.

Parte de la popularidad de los algoritmos de recomendación basados en *collaborative filtering* se pueden deber al Netflix Prize[5], competición que ofreció en 2006 un millón de dólares al algoritmo que mejor predijera la clasificación de los usuarios de películas basándose en clasificaciones previas, sin ninguna otra información.

Existe literatura diversa que trata sobre recomendadores de artículos científicos, aunque no se han detectado muchos casos donde se haya pasado de las pruebas de concepto a sistemas en producción. La mayoría de estos recomendadores son colaborativos [16]. Una aproximación común con métodos colaborativos son los que hacen *clustering* sobre materias [14], pues apuestan a que científicos de una misma área de conocimiento tienden estar interesados por los mismos artículos y basan sus recomendaciones en búsquedas anteriores de otros colegas de la misma área. Existen otros ejemplos similares [19, 20] que se basan en ontologías o taxonomías como la de ACM[6] y algunos [17, 29] en *tags* generados en plataformas como CiteSeer[7].

También existen otros recomendadores de artículos de tipo híbrido ([15, 21, 22, 23]), que combinan filtrado colaborativo y de contenido. Por ejemplo, existen aproximaciones usando citaciones [24], o basados en grafos [18].

4.1 Nuestra aproximación

Como detallamos en el anterior apartado, existe mucha literatura sobre recomendadores aplicada a artículos científicos. También existe algún precedente [10] de introducir arquitecturas de *big data* a los recomendadores, pero como en la mayoría de casos de recomendación de artículos, siguen la aproximación de filtrado colaborativo en vez de por contenidos. También hay algún ejemplo [26, 27] de introducir recomendadores de artículos en repositorios digitales, pero son parciales ya que se basan sólo en los contenidos de los propios repositorios, sin introducir en el algoritmo datos de otros repositorios externos.

[5] `https://en.wikipedia.org/wiki/Netflix_Prize`
[6] `http://www.acm.org/about/class`
[7] `http://citeseerx.ist.psu.edu`

El proyecto consiste en **aprovechar las arquitecturas actuales de big data para poder realizar un recomendador de artículos científicos de alcance global**. Como fuente de datos se quieren recolectar metadatos de todos los repositorios mundiales de acceso abierto.

Es un recomendador basado en contenido, concretamente se extraen de cada artículo unas palabras clave "representativas" que sirven para compararse con otros artículos y obtener las recomendaciones por "similitud".

5 Modelo de negocio

El proyecto distingue claramente el usuario del producto y el cliente. El recomendador es útil a los usuarios de repositorios digitales, por ejemplo, investigadores. Su propuesta de valor es poder acceder cuando consulten un artículo a otros similares y relevantes que pueden estar localizados en cualquier repositorio mundial, con el consecuente ahorro de tiempo de búsqueda y la posibilidad de acceder a contenidos que sin la recomendación podrían no haber encontrado.

Los clientes serán los propios repositorios digitales, quienes a través de un **modelo de suscripción**, podrán acceder a una API para poder ofrecer recomendaciones a nivel de artículo. Aquellos repositorios que dispongan de un buen recomendador a nivel de artículo serán mejor valorados y útiles a sus usuarios, valor añadido (propuesta de valor) respecto a aquellos que no dispongan de este servicio.

Un referente que inspira el modelo de negocio del proyecto es el de una *data science startup* como Altmetrics[8], que aún diferenciándose del producto, comparten el objetivo de ampliar el abanico de herramientas y recursos de los repositorios para hacer más enriquecedora la experiencia en ellos, agregando conocimiento de diferentes fuentes.

Para describir el modelo de negocio se usan dos marcos de referencia que son complementarios. Primero el *Business Model Canvas* [49] creado por Alexander Osterwalder, que actualmente es tendencia en proyectos que siguen la filosofia Lean Startup [50] iniciada por Eric Ries. El segundo es el *data-driven business-model framework* (DDBM) elaborado por Pattrick Max Hartmann, et altri [52], una taxonomia de modelos de negocio adaptada a *start-ups* de contenidos relacionados con *big data*.

5.1 *Business Model Canvas*

El modelo consiste en una plantilla donde representar gráficamente la propuesta de valor de una empresa, su infraestructura, sus clientes y sus finanzas. Consta de nueve piezas para describir un modelo de negocio.

A continuación se detallan los elementos del proyecto que se incluyen en cada parte del lienzo para describir el modelo de negocio. En la figura 5.1 se muestra el lienzo con estos conceptos relacionados entre sí con diferentes colores.

[8] `http://www.altmetric.com`

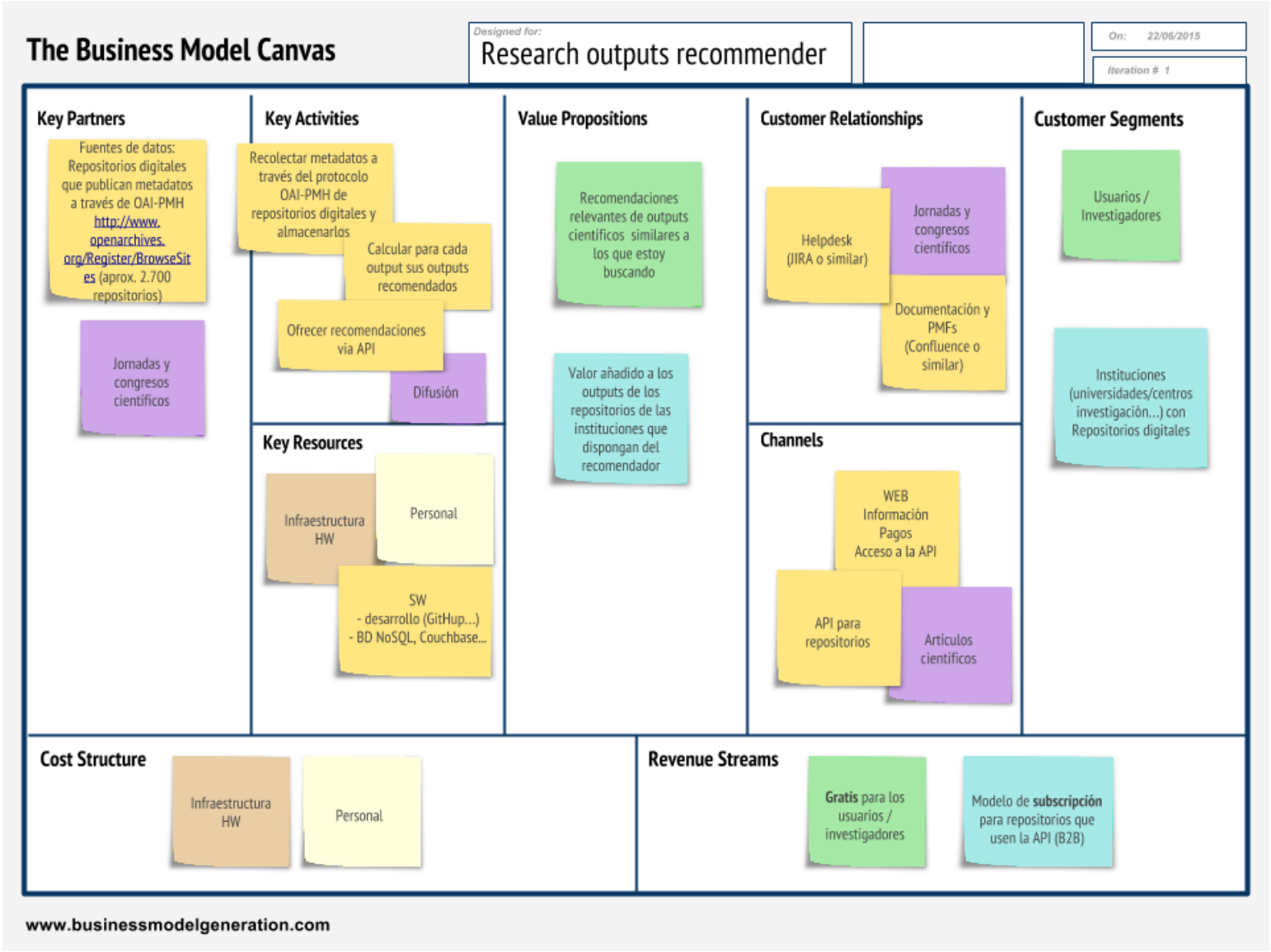


Figure 3: Business Model Canvas

1. Propuesta de valor

    - Para el usuario-investigador: recomendaciones relevantes de *outputs* científicos similares a los que estoy buscando.
    - Para el cliente-institución: valor añadido del repositorio de la institución, las recomendaciones serán útiles a sus usuarios y les hará más "atractivo" el repositorio.

2. Segmentos de clientes

    - Usuarios / Investigadores
    - Instituciones (universidades, centros de investigación...) que dispongan de repositorios digitales.

3. Canales

    - Web informativa, con información de acceso a la API, pagos, etc.
    - API de accesso a los repositorios institucionales clientes.
    - Artículos científicos sobre la herramienta

4. Relaciones con los clientes

    - Helpdesk, sistema Jira[9] o similar para gestionar las peticiones de los clientes.
    - Documentación y preguntas más frecuentes (FAQs), sistema Confluence[10] o similar.
    - Asistencia a jornadas y congresos.

5. Actividades clave

    - Recolectar metadatos periódicamente a través del protocolo OAI-PMH de repositorios digitales y almacenarlos.
    - Calcular para cada *output* sus *outputs* recomendados.
    - Ofrecer via API las recomendaciones.
    - Difusión

6. Recursos clave

    - Personal (generará coste después del prototipo, y de las horas asignadas al postgrado)
    - Infraestructura *hardware*
    - *Software* (HBase, Couchbase...)

[9] `https://www.atlassian.com/software/jira`
[10] `https://www.atlassian.com/software/confluence`

7. Socios clave

    - La tecnología OAI-PMH y su uso extendido entre la comunidad de repositorios digitales es clave pues se trata de la única fuente de datos usada.
    - Jornadas y eventos científicos donde poder hacer difusión del producto.

8. Costes

    - Infraestructura hardware.
    - Personal

9. Fuentes de ingresos

    - Para el usuario-investigador: gratis.
    - Para el cliente-institución: modelo de suscripción (B2B).

El apartado de costes se detalla en el anexo **??**.

5.2 *Data-driven Business-model Framework*

Este segundo marco tiene algunos puntos solapados con el modelo anterior, como el tipo de clientes o las fuentes de ingresos, pero incorpora alguno nuevo relacionado con el *big data* como el tipo de fuentes de datos y en otros puntos detalla más las opciones a escoger, por ejemplo, con siete tipos de actividades clave a escoger. Básicamente, ver si el modelo de negocio del proyecto encaja en esta taxonomía significa que está alineado con otras 300 *startups* del negocio de los datos.

En la figura 4 se muestra el data-driven business model y en él se detalla el encaje del proyecto.

En base a esta taxonomía, el marco también define seis tipologías de empresas combinando en una matriz los ejes de las fuentes de datos y las actividades clave. En la figura 5 se muestra la matriz. De las seis empresas tipo propuestas por el modelo, el proyecto encaja totalmente con el tipo A de empresa (*free data collector and aggregator*) y parcialmente con el D (*free data knowledge discovery*). Un ejemplo de empresa tipo A es Gnip[11], que combina datos abiertos de diferentes redes sociales para distribuirlos en forma de API. La diferencia con el proyecto es la tipología de los datos, que son *outputs* científicos en vez de social media.

6 Descripción de los datos

La fuente de datos del proyecto son los repositorios digitales, donde se encuentran depositados artículos científicos descritos con metadatos estructurados de una manera homogénea en el formato Dublin Core.

[11] `https://gnip.com/`

Figure 4: Data-driven business-model framework (DDBM)

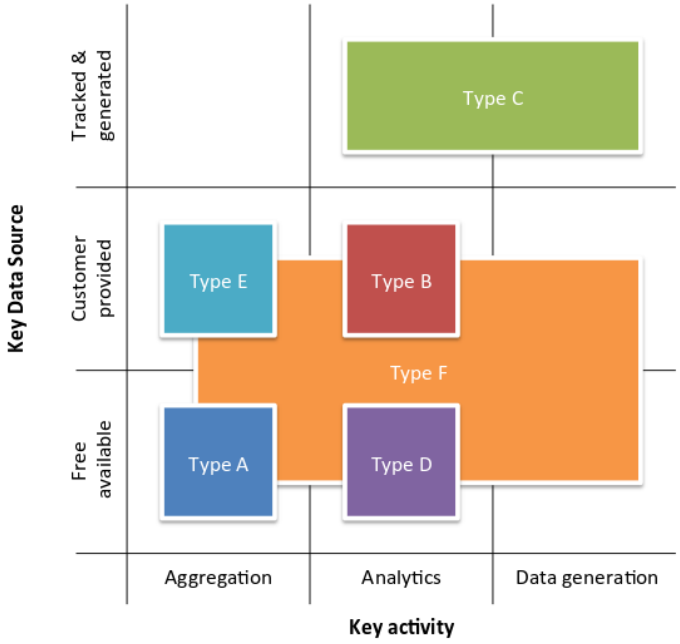


Figure 5: Tipus de data-driven start-ups en DDBM)

6.1 Estructura de los datos (Dublin Core)

Dublin Core (DC) es un sistema de 15 definiciones descriptivas que pretenden transmitir un significado semántico. Son opcionales, se pueden repetir y pueden aparecer en cualquier orden. Este sistema de definiciones fue diseñado para proporcionar una descripción básica de cualquier recurso digital (por ejemplo, artículos científicos) sin que importe el formato de origen, el área de especialización o el origen cultural. A continuación se detallan los 15 metadatos agrupados según el ámbito de información que contienen.

- Elementos relacionados con el contenido del recurso:
  - dc.title: nombre del recurso
  - dc.subject: clave del recurso, se fomenta el uso de vocabularios controlados y de sistemas de clasificación formal
  - dc.description: descripción textual del recurso. Puede ser un resumen (abstract) o una descripción del contenido.
  - dc.source: fuente del recurso
  - dc.language: lengua del recurso
  - dc.relation: relación del recurso con un identificador de otro recurso

  - dc.coverage: cobertura espacial y/o temporal del recurso. La cobertura espacial se refiere a una región física, indicada mediante coordenadas; y la cobertura temporal se refiere al contenido del recurso (no confundir con la fecha de creación del recurso, indicada con otro metadato)

- Elementos relacionados con la propiedad intelectual del recurso:
  - dc.creator: persona u organización responsable de la creación del recurso
  - dc.publisher: editor, entidad responsable de que el recurso se encuentre disponible en la red en su formato actual
  - dc.contributor: otras personas u organizaciones que hayan realizado una contribución intelectual significativa pero que sea secundaria en comparación con dc.creator

- Elementos relacionados con la instancia del recurso:
  - dc.date: fecha en la que el recurso se puso a disposición del usuario en su formato actual
  - dc.type: categoría del recurso, por ejemplo, si se trata de una página personal, un artículo científico, un poema, etc
  - dc.format: formato de datos del recurso, usado para identificar el *sottware* del recurso y, posiblemente, el hardware que se necesita para mostrar el recurso
  - dc.identifier: identificador unívoco del recurso. Puede tratarse de, por ejemplo, URL, URN, ISBN o *handlers*

En el anexo A se muestra, a modo de ejemplo, un fichero XML de un artículo científico del repositorio PubMed Central estructurado en formato Dublin Core.

6.2 Calidad de los datos

La calidad de los datos se puede medir por su estructura o por su contenido. Desde el punto de vista de la estructura se trata de una fuente de datos ideal, comparada con otras fuentes de datos abiertas, pues dispone de una estructura formal homogénea. La "V" de variabilidad de las fuentes de datos no aplicaría en este proyecto.

Desde el punto de vista del contenido, existen diversos riesgos y sus asunciones correspondientes a tener en cuenta, como por ejemplo:

- Duplicados: es posible que un mismo artículo esté depositado en más de un repositorio con identificadores diferentes. En este caso, es posible que pueda haber artículos repetidos en las recomendaciones. De echo, si recomienda "muy bien" es posible que pueda usarse, como se describe en la posterior sección de trabajo futuro, como una herramienta de detección de artículos duplicados. Se asume que en la primera versión de la herramienta se recomendaran duplicados, en versiones posteriores, se

puede desarrollar un algoritmo de selección del artículo más "representativo" entre los posibles duplicados.

- Artículos sin resumen (*abstract*): como no es obligatorio el metadato dc.description, es posible que no exista en algunos artículos. Como el *abstract* tiene un peso muy importante en el recomendador, se asume que aquellos artículos que no dispongan de resumen tendrán menos posibilidades de ser recomendados a sus artículos "similares" teóricos.

- No existe una clasificación universal de materias usada de manera homogénea en diversos repositorios. En algunos se usa el código Unesco[12], en otros la clasificación decimal universal (CDU)[13], etc. Esto impide que en el recomendador se puedan hacer *clusterings* precisos que beneficiarían el rendimiento. Se asume que como el carácter del proyecto es abordar a los repositorios de manera global, mientras no exista un clasificador usado por la mayoría de fuentes de datos, no se podrán aplicar técnicas de *clustering* en la recomendación.

- *Sparsity*: para hacer las recomendaciones se compararan las palabras clave de los artículos entre si. Al disponer de millones de artículos donde se comparan "pocas" palabras, seguramente la superposición de artículos con palabras muy similares será complicada. Para disminuir esta sparsidad de los datos, se ha introducido en el preprocesado la comparación con tesaurus de materias para encontrar palabras clave en los resumenes y títulos de los artículos. De esta manera, se espera incluir materias "normalizadas" en las palabras clave y aumentar de este modo las posibles similitudes entre artículos de la misma área de conocimiento.

7 Arquitectura del sistema

En la figura 6 se muestra la arquitectura de la aplicación, que se detalla módulo a módulo a continuación.

7.1 Módulos

El sistema se divide en cuatro módulos diferenciados: el de la obtención y almacenamiento de los datos (*inputs*), el de su preprocesado, el del proceso de realizar recomendaciones y el de ofrecer un acceso a los datos resultantes, las recomendaciones (*outputs*).

7.1.1 Obtención y almacenamiento de los datos (*inputs*)

El módulo de obtención de datos implementa un recolector OAI-PMH[14] para extraer metadatos de repositorios digitales, como PubMed Central, y los almacena en un *wide-column store*

[12] `https://es.wikipedia.org/wiki/Clasificaci\%C3\%B3n_Unesco`
[13] `https://es.wikipedia.org/wiki/Clasificaci\%C3\%B3n_Decimal_Universal`
[14] `https://www.openarchives.org/pmh`

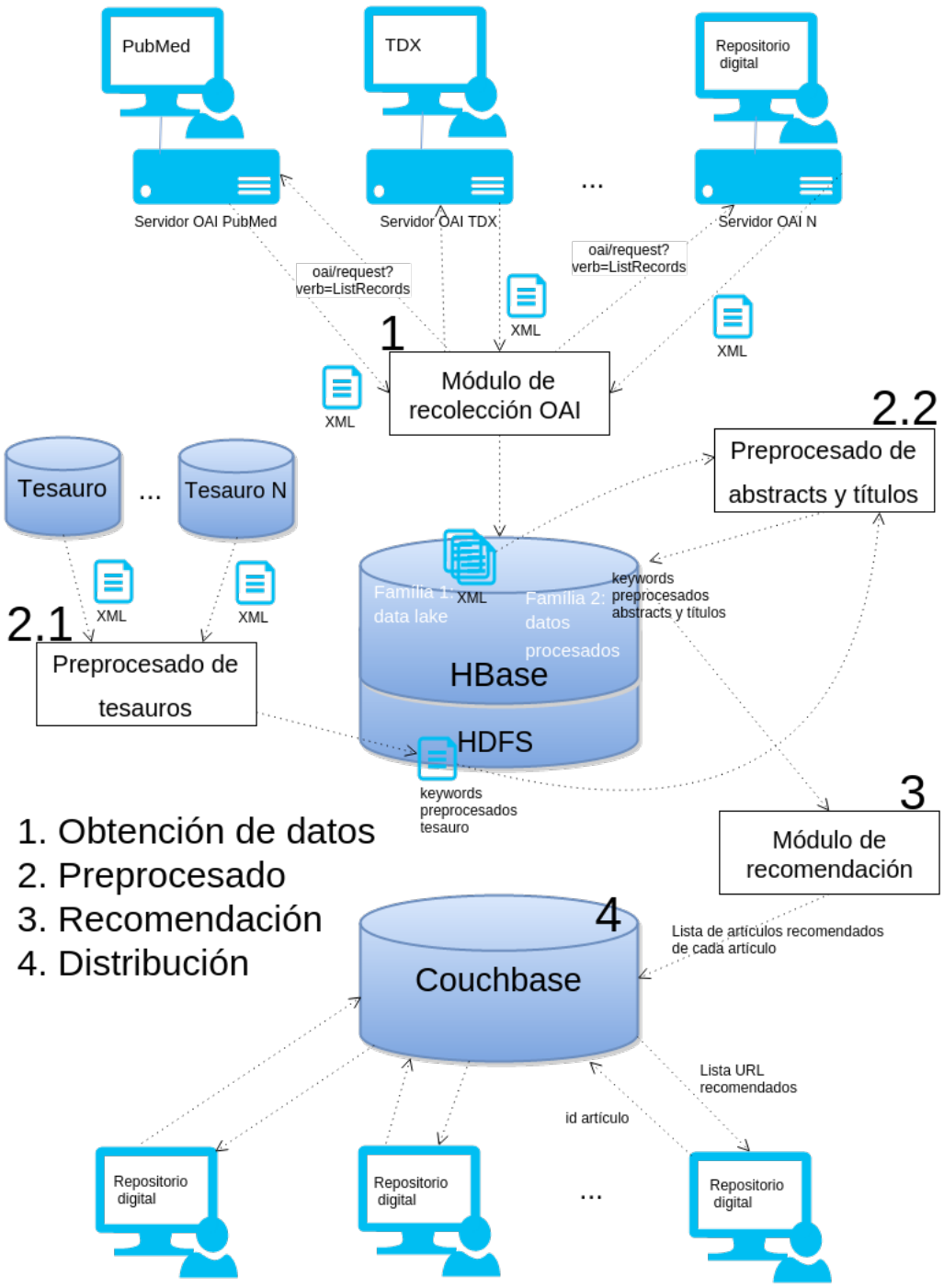


Figure 6: Arquitectura

(HBase).

Los recolectores de metadatos son servicios basados en la interoperabilidad entre diferentes repositorios digitales de contenidos digitales. Su ADN está formado por el protocolo OAI-PMH (Protocol for Metadata Harvesting de la Open Archives Initiative). Se basa en que los documentos residan en los repositorios, mientras que sus metadatos y su correspondiente enlace se desplacen a través de los diferentes proveedores que ofrecen servicios de valor añadido mediante diferentes agregaciones de estos recursos distribuidos.

Por ejemplo, las tesis doctorales de las diferentes universidades españolas se encuentran depositadas en repositorios institucionales de las propias universidades o en repositorios cooperativos como Tesis Doctorales en Red (TDR)[15]. Mediante OAI-PMH, existe un recolector que permite una consulta de manera agregada a nivel nacional, y otro, llamado DART[16], que las agrupa a nivel europeo.

En la compartición de metadatos participan dos actores, los proveedores de servicios (*service providers*) y los proveedores de datos (*data providers*). Si un repositorio ofrece sus datos a través del protocolo es un proveedor de datos. Por otro lado, un repositorio puede recolectar metadatos de otros repositorios, convirtiéndose así en un proveedor de servicios y ofreciendo un servicio de valor añadido.

A nivel técnico, el protocolo utiliza transacciones HTTP para emitir preguntas y obtener resultados entre un proveedor y un servidor de metadatos. El segundo le puede pedir al primero que le envíe metadatos según determinados criterios, como por ejemplo, la fecha de creación de los recursos. Como respuesta, el segundo retorna un conjunto de registros en formato XML como el detallado en el apartado previo de descripción de los datos.

Existen seis peticiones que un cliente puede realizar a un servidor, que se emiten usando los métodos GET o POST del protocolo HTTP:

- GetRecord: permite obtener un documento del repositorio dado su identificador unívoco.
- Identify: permite la identificación de los parámetros básicos del repositorio. Generalmente se utiliza como puerta de entrada a la interfaz OAI y para comprobar su validez.
- ListMetadataFormats: permite saber que formatos de datos exporta el repositorio a través de OAI. Generalmente todos soportan la exportación en formato Dublin Core.
- ListRecords: permite la obtención de un conjunto de documentos, ya sea del repositorio entero o de un set especificado. Permite la utilización de un parámetro de fecha, en el cual podemos especificar desde cuando queremos obtener los documentos, pudiendo realizar recolecciones incrementales y así obtener los documentos añadidos o modificados en un día.

[15] `http://www.tdx.cat`
[16] `http://www.dart-europe.eu`

- ListIdentifiers: es una forma abreviada del verbo ListRecords. Permite obtener las cabeceras de los documentos en lugar del documento entero.
- ListSets: permite la obtención del listado de sets (agrupaciones de documentos) que exporta el repositorio a través de OAI.

Un ejemplo de llamada que retorna el XML mostrado en el apartado previo de descripción de datos es: `http://www.pubmedcentral.nih.gov/oai/oai.cgi/request?verb=ListRecords&metadataPrefix=oai_dc&from=2000-01-01`

Este XML puede ser pretratado para eliminar toda la información que no será usada en la recomendación, pero se ha decidido almacenar los datos en crudo en una base de datos (HBase) a modo de *data lake* por su utilidad en futuras aplicaciones más allá del actual proyecto de recomendación.

7.1.2 Preprocesado de los datos

Los datos almacenados en HBase en el proceso anterior necesitan ser procesados para facilitar la labor del módulo de recomendación. Se necesita disponer de una serie de conceptos (*keywords*) representativos del contenido del documento. Estos se encuentran en los metadatos de materias (dc.subject), título (dc.title) y resumen (dc.description).

El procesado de las materias es directo, pues en el XML original ya se encuentran separadas en diferentes *tags*, pero tanto el título como el resumen son textos. En lugar de que el recomendador compare textos, y entrar en el ámbito del lenguaje natural[17], se realiza un proceso de extracción de *keywords* del título y el resumen de los artículos.

Uno de los pasos de esta extracción de palabras clave del resumen y el título es comparar si estas se encuentran en algún tesauro especializado, pues se cree que dichas palabras serán representativas del texto, y serán buenas candidatas de cara a la recomendación. Si ninguna de las palabras se encuentra en los tesauros, se considera que las más representativas serán aquellas que aparecen con más frecuencia.

Potencialmente existen muchos tesauros contra los que poder comparar las palabras de los *abstracts*, como prueba de concepto se ha implementado con uno. El seleccionado ha sido Medical Subject Headings (Mesh)[18], el vocabulario controlado usado para la indexación de contenidos en PubMed.

Tanto a los tesauros como a los *abstracts* se les aplica el mismo preprocesado para que puedan ser comparados, incluyendo una limpieza de *stop words*[19] en inglés y castellano y su stemización[20] [31]. En el pseudocódigo 1 se muestra el procesado de los tesauros, que se realiza una única vez para cada tesauro (o cada vez que se actualice); y en el pseudocódigo 2 se detalla el preprocesado de los *abstracts* y títulos de los artículos científicos.

[17] `http://www.talp.upc.edu`
[18] `http://www.ncbi.nlm.nih.gov/mesh`
[19] `https://en.wikipedia.org/wiki/Stop_words`
[20] `https://en.wikipedia.org/wiki/Stemming`

Los *keywords* resultantes del preprocesado son almacenados en HBase en una familia diferente de la del XML original para facilitar el acceso separado desde el módulo de recomendación.

Algoritmo 1 Preprocesado de un tesauro

Entrada: Tesauro (en XML). del XML del tesauro
Salida: Listado de palabras clave preprocesadas del tesauro (.TXT en HDFS)
1: Leer elementos del XML del tesauro (librería JAXB[21])
2: **mientras** quedan líneas por procesar **hacer**
3: Borrar elementos XML
4: Pasar palabras a minúsculas
5: Borrar stop words, acentos y caracteres que no sean letras
6: Stemizar las palabras
7: Escribir las palabras resultantes en un .TXT en HDFS
8: **fin mientras**

7.1.3 Recomendaciones

El proyecto es un recomendador de contenido donde **se calculan distancias lógicas entre las palabras clave de los diferentes documentos**.

Para comparar documentos se usan tanto *keywords* originales, como las materias (dc.subject), como todos aquellos resultantes del preprocesado del *abstract* y el título del documento. Se considera que la comparación de estos keywords de diferentes documentos puede dar lugar a recomendaciones de documentos "similares".

Por simplicidad, para esta primera versión del algoritmo no se ha considerado el uso de metadados relacionados con la fecha de publicación de los artículos (dc.date). De todos modos, existen estudios que destacan la componente temporal en la recomendaciones [6], ya que la ciencia va avanzando a hombros[22] de trabajos previos, y un artículo reciente de un tema seguramente estará más alineado a los intereses de los investigadores. Como se menciona en un apartado posterior, la introducción de la componente temporal en el recomendador será analizada en futuros trabajos.

En la recomendación, existen dos componentes básicos a tener en cuenta:

- Similaridad: tipo de distancia lógica entre las palabras clave de dos artículos. En los artículos de la bibliografía muchos de los recomendadores usan la distancia de Jaccard[23] o la del coseno[24].

- Técnica usada para calcular las distancias: una aproximación del tipo "fuerza bruta" se realiza si se calcula la distancia de cada artículo a todos los demás para después

[22] Cita de Issac Newton: "Si he logrado ver más lejos, ha sido porque he subido a hombros de gigantes"
[23] `https://en.wikipedia.org/wiki/Jaccard_index`
[24] `https://en.wikipedia.org/wiki/Cosine_similarity`

Algoritmo 2 Preprocesado de *abstracts* y títulos de artículos científicos

Entrada: Metadatos de artículos científicos (en XMLs en HBase)
Salida: Listado de palabras clave preprocesadas del artículo (en otra família de HBase)

```
1: Leer del HBase los XML con los metadatos de los artículos
2: para cada artículo hacer
3:    Seleccionar y concatenar el abstract (dc.abstract) y el título (dc.title)
4:    Pasar palabras a minúsculas
5:    Borrar stop words, acentos y caracteres que no sean letras
6:    Stemizar las palabras
7:    Contar la frecuencia de las palabras
8:    Comprobar si cada palabra se encuentra en el tesauro
9:    NumKeywords = 10;
10:   si NumPalabras menor que NumKeywords entonces
11:      Escribir en HBASE null
12:   si no
13:      si Num. palabras en tesauro mayor que NumKeywords entonces
14:         Escribir en HBase las NumKeywords palabras más frecuentes que están en el
            tesauro
15:      si no, si Núm. palabras en tesauro = 0 entonces
16:         Escribir en HBase las NumKeywords palabras más frecuentes
17:      si no
18:         Escribir en HBase las palabras encontradas en el tesauro y completar hasta
            NumKeywords con las más frecuentes
19:      fin si
20:   fin si
21: fin para
```

almacenar los N más similares. Existen otras aproximaciones que para escoger estos N artículos similares que usan probabilidades y otras técnicas para no haber de calcular exhaustivamente todas las distancias, con el consecuente ahorro en costes de computación.

Implementar un método para computar las distancias de manera eficiente estaba fuera del alcance del prototipo, y por eso se ha buscado alguno ya existente que cumpliera con los requisitos del proyecto. La respuesta ha sido DIMSUM [11, 12], un algoritmo desarrollado por Twitter e integrado el la librería MLib de Spark (a partir de la versión 1.2). Permite hacer comparativas entre registros de manera eficiente usando probabilidades.

Finalmente, ¿como saber si el recomendador del prototipo recomienda "bien"? Están documentadas diversas técnicas, con diferentes costes, para poder verificar los resultados de estas herramientas [1], la mayoría se basan en la comparación "humana" entre dos configuraciones (tests A/B[25]).

Por ejemplo, en [25] realizan una comparación entre los resultados de cuatro algoritmos de recomendación diferentes, y es una muestra de 150 personas "expertas" quienes validan qué aproximación cumple mejor su cometido.

En el prototipo, se realizarán comprobaciones manuales en una muestra muy reducida de usuarios para intuir el correcto criterio de recomendación del prototipo. Como se indica en un apartado posterior, quedará para futuros trabajos modificar alguna configuración e ir haciendo iteraciones guiadas por las opiniones de usuarios.

7.1.4 Acceso a las recomendaciones (*outputs*)

Del anterior proceso de recomendación se obtiene un listado de documentos recomendados por cada documento de la base de datos HBase.

Esta información se podría almacenar en otra familia de HBase, pero se ha escogido almacenar los resultados en una caché de vistas (Couchbase) que se adapta mejor a los requisitos de este módulo. Principalmente por:

- Rápido acceso a los datos.
- Frecuencia de actualización: las recomendaciones son generadas periódicamente con una frecuencia que se estima elevada (diaria o semanal).
- Posible pérdida de datos: en este caso, los datos se pueden volver a computar.

La clave de la vista es el identificador unívoco del documento y como valor el listado de recomendados, formado por los identificadores de cada uno de los documentos más "similares". De este modo, para el prototipo, la interfaces web de Couchbase hacen las funciones de API, y es consultada desde los diferentes repositorios para, dado un documento, obtener todas sus recomendaciones.

[25] `https://en.wikipedia.org/wiki/A/B_testing`

7.2 Esquema de la base de datos

La base de datos HBase dispone de dos familias principales. Una está formada por una única columna donde se almacena el XML original de los metadatos de cada documento (ver figura 7) y otra familia donde se almacenan las diferentes palabras clave (ver figura 12)que son posteriormente utilizadas en el algoritmo de recomendación. El identificador de cada documento se usa como clave.

La caché de vistas Couchbase dispone de una vista con los pares clave-valor formados por el identificador del documento y una lista con los identificadores de los documentos más parecidos (ver figura 15).

7.3 Flujo de datos

Existen cuatro flujos de datos en el sistema: la recolección de los datos, su preprocesamiento, la comunicación entre el proceso de recomendación y Couchbase y cómo los repositorios digitales (clientes) podrán acceder a Couchbase.

1. En el primero se realizan peticiones HTTP a las diferentes interfaces OAI y éstas devuelven respuestas en formato XML con datos estructurados. Para la obtención de todos los registros de un repositorio será necesaria, generalmente, más de una petición, puesto que por cada petición se obtienen un número finito de documentos. Para ello se hace uso del valor "resumptionToken", que apunta a los diferentes sets que tenga el repositorio, permitiendo implementar esta comunicación como un proceso automatizado. Estos datos recolectados se almacenan en HBase.

2. En el segundo flujo de datos hay una interacción entre HBase, que contiene los datos en crudo, y Spark, que limpia y preprocesa los textos del abstract y del título en palabras clave que se vuelven a almacenar en (otra) de HBase.

3. El tercer flujo de datos parte de los datos preprocesados almacenados en HBase para realizar el algoritmo de recomendación. El resultado (las recomendaciones) se almacena en Couchbase.

4. El último flujo de datos se produce entre Couchbase y los repositorios clientes. Estos, mediante peticiones HTTP a la interfaz web de Couchbase, harán consultas a la vista definida y obtendrán un listado de identificadores (URL) para mostrar al usuario las recomendaciones. Es el paso inverso al primer flujo de datos, donde se solicitaban datos a los repositorios y éstos los facilitaban a través de OAI-PMH; ahora son los repositorios quienes solicitan datos de la aplicación a través de Couchbase.

8 Selección de herramientas

Se ha escogido el ecosistema Hadoop para la realización del proyecto. Aunque por el volumen de información del prototipo, con "sólo" un par de repositorios, sería posible haber escogido

un *document store*, como MongoDB, u otras herramientas, se ha seleccionado el ecosistema Hadoop por su potencial escalabilidad. El alcance del prototipo es limitado, las expectativas de crecimiento son ambiciosas porque el alcance del proyecto son todos los repositorios digitales mundiales que ofrecen sus datos en abierto por el protocolo OAI-PMH (más de 3.000[26]).

Las herramientas concretas escogidas para el desarrollo del prototipo son el *wide-column store* HBase para el almacenamiento de los datos crudos y preprocesados (en familias diferentes) montado sobre un sistema de ficheros distribuido HDFS; Spark para el preprocesado de datos y de base para realizar las recomendaciones y finalmente Couchbase como base de datos de rápido acceso para almacenar las recomendaciones. A continuación se justifica la elección de estas herramientas.

- HBase. Su gestión de versiones e incrementales se adapta perfectamente a la estructura de datos. Los metadatos de los artículos en origen pueden variar y se pueden volver a recolectar. Como se usa como clave de los artículos su identificador unívoco, al recolectarse de nuevo la base de datos facilita la gestión de actualización enfrente de un sistema de ficheros como puede ser HDFS.

  También se ha valorado HDFS para guardar las palabras clave procesadas porque al final necesitaremos acceso secuencial a todos los registros para hacer la recomendación, pero finalmente se ha escogido HBase para aprovechar la estructura de familias que ofrece. Por otro lado, fuera del alcance de este proyecto, y como se detalla en el apartado de trabajo futuro, se puede aprovechar el acceso por valor de HBase para ofrecer servicios alternativos sobre los mismos datos (o con preprocesados diferentes en otra familia), por ejemplo, para dados dos artículos comprobar si son duplicados.

- Spark. Buen encaje con el resto de herramientas. Existen funciones en la librería MLIB para el preprocesado textual de los datos (contador de frecuencias de palabras, borrado de stop-words...).

- Couchbase. Su naturaleza de caché de vistas encaja con la necesidad de ofrecer rápido acceso a las recomendaciones. Además, los datos almacenados en este sistema no son críticos. Su pérdida no implica comenzar el proceso de nuevo porque los datos fuente (en crudo y preprocesados) están almacenados en HBase.

9 Análisis cuantitativo

Uno de los objetivos del prototipo es obtener datos cuantitativos de volumen y especialmente de rendimiento para poder realizar estimaciones del sistema global y analizar su viabilidad.

El prototipo se realizará con datos de un par de repositorios, pero el sistema completo tiene un potencial de más de 3.000. Analizando métricas de volumen y tiempo, en cada fase de las mencionadas anteriormente, se podrán detectar cuellos de botella y obtener datos

[26] //TODO posar referencia a llistat de data providers

estadísticos del funcionamiento del sistema, por ejemplo, el tiempo medio necesario de carga por artículo, o el tiempo de cómputo necesario para realizar las recomendaciones (1 hora, 3 días...) en base a la infraestructura usada y el volumen del sistema.

9.1 Rendimiento

A continuación se detallan las principales métricas que se quieren obtener en cada módulo del prototipo, y se analiza la importancia de sus costes temporales.

- En la recolección de datos inicial: el coste temporal de recolección inicial puede ser elevado, pero no se estima que sea un problema pues se trata de un tratamiento en modo *batch* que se realizará una sola vez para cada repositorio del sistema. Es un proceso que si se quiere acelerar se puede paralelizar fácilmente por repositorio aumentando recursos *hardware* hasta que finalice, aprovechando la elasticidad de sistemas de *cloud computing*. Las métricas que se quieren obtener son las siguientes:
  - Tiempo (en segundos) de recolección de cada repositorio
  - Espacio (en MB) necesario para almacenar cada repositorio
  - Tiempo medio de recolección de un artículo
  - Espacio necesario para almacenar un artículo
  - Número medio de artículos por repositorio
- En la recolección de datos incremetal: para actualizar las recomendaciones con los nuevos artículo los que vayan surgiendo se realizaran recolecciones incrementales. A través del protocolo OAI-PMH se obtendrán metadatos de artículos nuevos o que se hayan modificado en los repositorios. Se estima que se realizarán pocas modificaciones y muy pocas eliminaciones de contenido, pues la naturaleza de los repositorios es incremental. Esto significa que las métricas del apartado anterior sirven para hacer estimaciones del coste de estas recolecciones, a las que se les puede añadir:
  - Número de artículos nuevos que se añaden a un repositorio en un periodo de tiempo determinado (métrica teórica)
- En el preprocesado de datos: el coste temporal del preprocesado puede llegar a ser alto dependiendo de los procesos que se realicen. Como en el punto anterior, no se estima que sea un problema pues se trata también de un proceso en *batch* que se realizará una única vez para cada artículo. Las métricas que se quieren obtener son las siguientes:
  - Tiempo medio de preprocesado del abstract de un artículo en keywords
  - Tiempo medio de preprocesado del título de un artículo en keywords
  - Tiempo medio de preprocesado de un tesauro

- En las recomendaciones: el coste temporal de este proceso puede ser elevado puesto que para obtener las recomendaciones de un artículo es necesario compararlo de manera secuencial con el resto de artículos. Por un lado, este coste no es un problema pues se trata también de un proceso en batch y no es necesario el tiempo real (como se muestra en el siguiente punto, mientras no se recalculen las recomendaciones, se puede acceder a las realizadas con anterioridad). Por otro lado, existe una clara relación del tiempo de cómputo con el volumen de artículos del repositorio. Conocer las métricas de este apartado es uno de los retos de este proyecto, porque si son elevadas se complica la viabilidad de poder escalar a todos los repositorios mundiales. La métrica que se quiere obtener es la siguiente:
  - Incremento del tiempo del proceso de recomendación en función del aumento del número de artículos del sistema.
- En el acceso a las recomendaciones: el coste de acceso via API a las recomendaciones de un artículo ha de ser muy pequeño. Con la arquitectura definida no se estiman problemas de rendimiento porqué en el proceso anterior se calculan las recomendaciones de todos los artículos. El coste temporal será el de la comunicación HTTP con el sistema, por esta razón, no se estima necesario obtener métricas de este proceso.

10 Resultados

En este apartado, para cada módulo del proyecto, se presentan los indicadores y métricas obtenidos en el prototipo y las dificultades que se han experimentado y que han hecho modificar previsiones iniciales.

El prototipo se ha realizado utilizando un ordenador portátil, en el anexo **??** se especifican las características del clúster con el que se podría realizar otra prueba de concepto disponiendo de más características *hardware* que podrían hacer mejorar la métricas que aquí se presentan.

10.1 Obtención y almacenamiento de los datos

Se ha logrado recolectar con éxito los metadatos de dos repositorios digitales, uno con las tesis doctorales de 17 universidades y otro con artículos de carácter médico. Concretamente:

- Número de artículos recolectados: 849.799 (20.316 de TDX y 849.799 de Pubmed)
- Espacio en HDFS: 1,8 GB
- Espacio necesario para almacenar un artículo (en media): 2,3 KB
- Tiempo de recolección de cada repositorio: pendiente.

Inicialmente estaba previsto usar un módulo de recolección de metadatos OAI-PMH que actualmente está implementado en repositorios digitales en producción, como el propio

TDX y del que los autores del proyecto ya disponían de experiencia previa. Dicho módulo recolecta metadatos y los almacena en ficheros XML.

Sin embargo, en vez de recolectar los XML en un sistema de ficheros y después volcarlos a HBase, se ha decidido hacer que a medida que se realiza la recolección se inserten directamente los XML en HBase para mejorar la eficiencia de la solución. Para ello, primero se optó por modificar el código (libre) del módulo, pero tras algún intento infructuoso finalmente se implementó de nuevo el recolector.

La nueva implementación ha resultado ser efectiva (850K artículos directamente a HBase), pero no del todo eficiente, pues se han producido diversos cortes en la recolección. Debugando en el problema se ha llegado a la conclusión de que en el anterior módulo se producía una mejor gestión de la conexión para evitar dichos cortes. Al tratarse de un prototipo se ha escogido la opción pragmática de tratar "manualmente" estos cortes en vez de mejorar la gestión de la conexión.

Debido a la existencia de estos cortes, no se ha podido disponer de una métrica temporal fiable de recolección de un repositorio. Después de un corte, se seguía recolectando desde el punto donde se había producido el error.

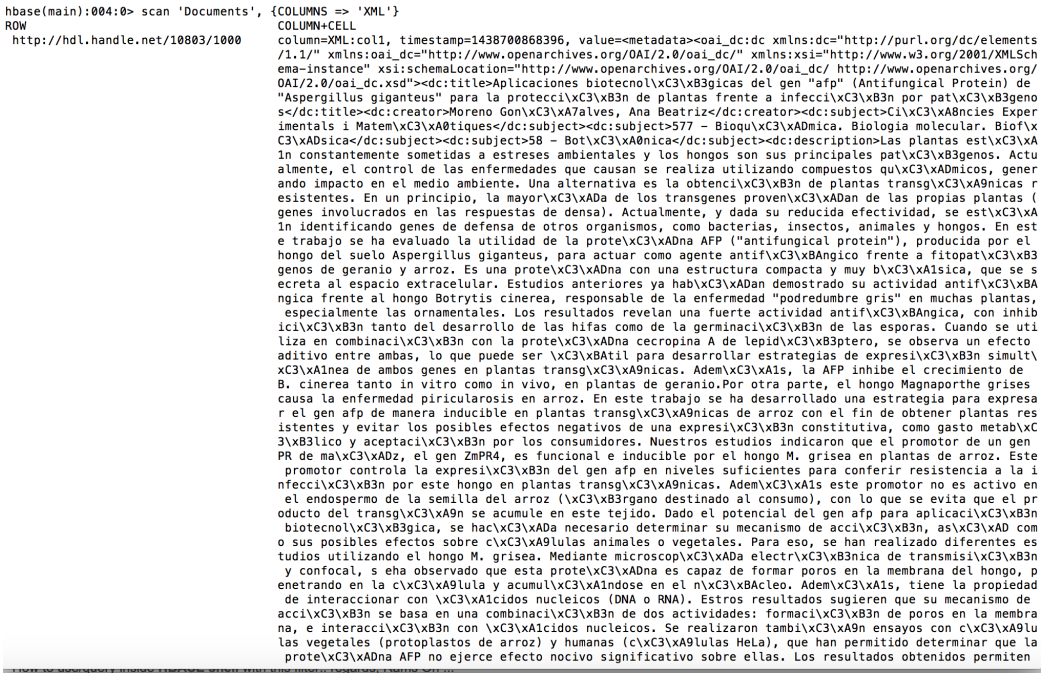


Figure 7: XML en HBase de un artículo recolectado

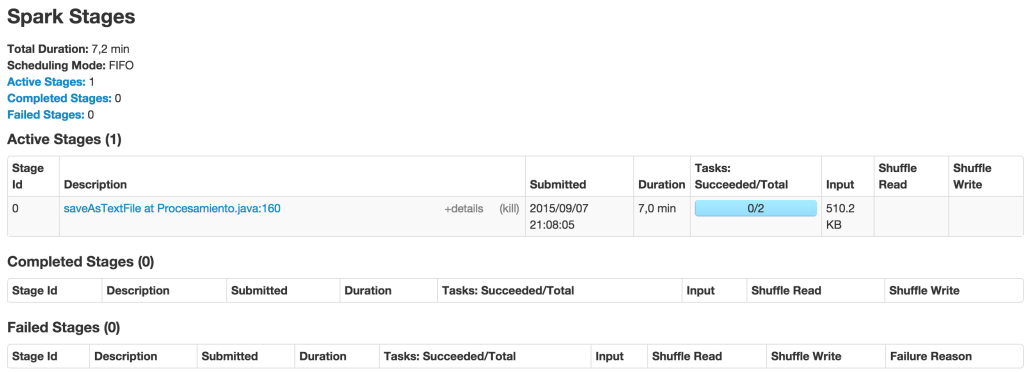
El proceso de recolección obtiene como resultado una familia en HBase en la tabla "Documents" donde se almacena, para cada artículo, el XML resultante de la recolección (ver figura 7. En esta aproximación se usa HBase como *data lake* pues almacenamos todo el XML aunque muchos de los metadatos del artículo no serán necesarios para el proceso

de recomendación. Si en el futuro, se necesitas otros metadatos para otros propósitos, ya no será necesario recolectar de nuevo.

Si el proyecto se estima viable y continúa su crecimiento con la recolección de nuevos repositorios digitales, se necesita dotar a esta nueva implementación de los métodos de gestión de la conexión que disponía el anterior módulo.

10.2 Preprocesado

Se ha logrado preprocesar con éxito el tesauro Mesh (ver figura 8 y 9) y almacenar el resultado en HDFS con las siguientes métricas:



Spark Stages

Total Duration: 7,2 min
Scheduling Mode: FIFO
Active Stages: 1
Completed Stages: 0
Failed Stages: 0

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
0	saveAsTextFile at Procesamiento.java:160 +details (kill)	2015/09/07 21:08:05	7,0 min	0/2	510.2 KB		

Completed Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write

Failed Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write	Failure Reason

Figure 8: Preprocesado del tesauro

```
08:06 INFO HadoopRDD: Input split: file:/Users/Ruben/Desktop/descriptors.txt:0+261208
08:06 INFO HadoopRDD: Input split: file:/Users/Ruben/Desktop/descriptors.txt:261208+261209
28:55 INFO FileOutputCommitter: Saved output of task 'attempt_201509072108_0000_m_000001_1' to hdfs://localhost:9000/hbase/resultats.tx
28:55 INFO SparkHadoopWriter: attempt_201509072108_0000_m_000001_1: Committed
28:55 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 1939 bytes result sent to driver
28:55 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 1249640 ms on localhost (1/2)
33:13 INFO FileOutputCommitter: Saved output of task 'attempt_201509072108_0000_m_000000_0' to hdfs://localhost:9000/hbase/resultats.tx
33:13 INFO SparkHadoopWriter: attempt_201509072108_0000_m_000000_0: Committed
33:13 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1939 bytes result sent to driver
33:13 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1507681 ms on localhost (2/2)
33:13 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
33:13 INFO DAGScheduler: Stage 0 (saveAsTextFile at Procesamiento.java:160) finished in 1507,692 s
33:13 INFO SparkContext: Job finished: saveAsTextFile at Procesamiento.java:160, took 1507.820163 s
33:13 INFO SparkUI: Stopped Spark web UI at http://192.168.1.36:4040
33:13 INFO DAGScheduler: Stopping DAGScheduler
33:14 INFO MapOutputTrackerMasterActor: MapOutputTrackerActor stopped!
33:14 INFO ConnectionManager: Selector thread was interrupted!
33:14 INFO ConnectionManager: ConnectionManager stopped
33:14 INFO MemoryStore: MemoryStore cleared
33:14 INFO BlockManager: BlockManager stopped
33:14 INFO BlockManagerMaster: BlockManagerMaster stopped
33:14 INFO SparkContext: Successfully stopped SparkContext
33:14 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.
33:14 INFO RemoteActorRefProvider$RemotingTerminator: Remote daemon shut down; proceeding with flushing remote transports.
33:14 INFO Remoting: Remoting shut down
33:14 INFO RemoteActorRefProvider$RemotingTerminator: Remoting shut down.
```

Figure 9: Consola final del procesado del tesauro

- Número de descriptores del Mesh: 27.455

- Espacio: 510 KB
- Tiempo de preproceso: 25 minutos

El preprocesado de los resúmenes y títulos de los artículos para extraer palabras clave que son útiles para la recomendación ha resultado más complicada. Las métricas de la primera implementación realizada eran las siguientes:

- Número de artículos preprocesados: 23K (de los 850K)
- Tiempo de preproceso: 12 horas

Se consideró demasiado bajo el rendimiento de la solución y se realizaron diversos cambios en la implementación con Spark del algoritmo hasta conseguir unos tiempos razonables.

En la primera aproximación se obtó por declarar diferentes *steps* dentro del *pipeline*, modularizando los diferentes procedimientos, sin tener en cuenta el número de objetos instanciados. En el último *step*, donde se almacenaban los resultados en HBase, quedo patente la ineficiencia del código al requerir demasiada memoria tanto por elevado número de instancias como de los diferentes bucles implementados.

Antes de centrar los esfuerzos en la modificación del algoritmo, se creyó que el bajo rendimiento era provocado por la escritura en HBase, pues antes de añadir el último *step* al *pipeline* el rendimiento era bueno. En este sentido, se usaron librerías externas como SparkOnHBase[27] que simplificavan la conexión entre Spark y HBase, però el rendimiento era inaceptablemente similar. También se modificó la configuración de HBase con la esperanza de encontrar un aumento de velocidad del procesado. Se desactivó la escrita a WAL, se aumentó el tamaño de los datos antes de hacer un flush para evitar un volcado constante, se modificó el pre-split de la tabla para favorecer la paralización con la función *foreachpartition* y, finalmente, se definió un modo pseudo-distribuido levantando diferentes RegionServers en una misma máquina (ver figura 10 para intentar explotar de esta manera más el paralelismo en la escritura.

Finalmente se entendió que no se avanzaba en la dirección correcta cuando se intento guardar los resultados a disco con la función *saveastextfile*. Cualquier acción que implicara almacenar datos fuera del contexto de ejecución de la aplicación estaba fuertemente penalizada por la poco eficiente implementación del procesado anterior.

En este punto, se modificó el algoritmo uniendo *steps* del *pipeline* para evitar el tiempo de ejecución de su creación, se instanciaron el máximo de objetos fuera del bucle principal y se revisaron las iteraciones para minimizar costes. También se aplico una decisión de diseño, limitar a 10 las palabras clave que se extraerían de cada artículo. Esta decisión, a parte de mejorar el rendimiento del preprocesado, también se tomó pensando en el siguiente módulo, de recomendación, pues para el prototipo una entrada fija de *keywords* facilitaba su posterior tratamiento matricial.

Con estos cambios se ha logrado un rendimiento aceptable (ver figura 11), concretamente:

[27] `https://github.com/tmalaska/SparkOnHBase`

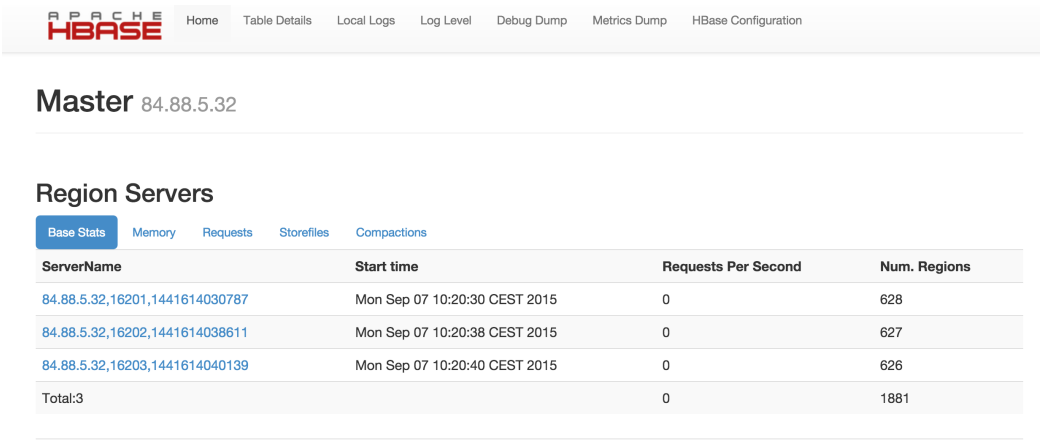


Figure 10: Diferentes RegionServers en un solo servidor

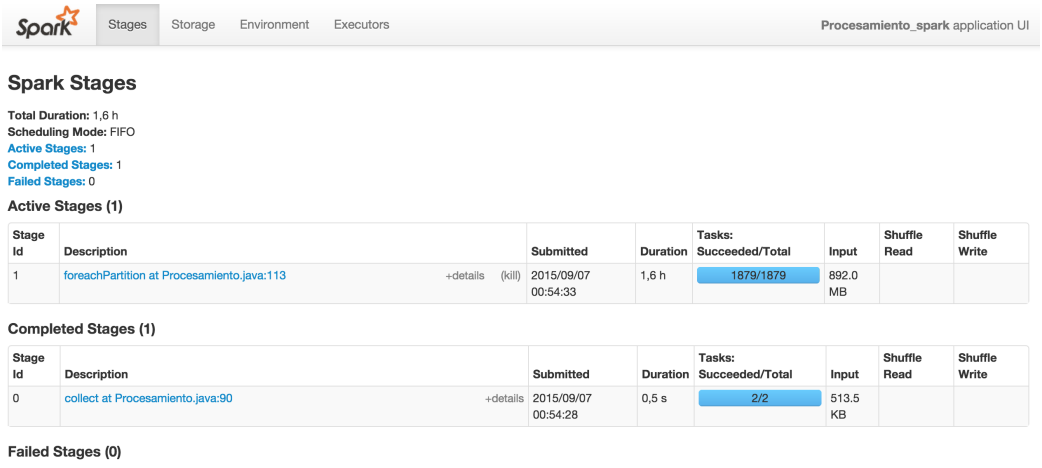


Figure 11: Preprocesado de los resúmenes y títulos de los artículos

- Número de artículos preprocesados: 850K (antes eran 23K)
- Tiempo de preproceso: 1,6 horas

Otro cambió que se decidió probar fue eliminar la parte de código de comparación de las palabras del abstract y el título con el tesauro. Cada palabra se comparaba con el tesauro hasta encontrar una coincidencia, en el peor de los casos (que es frecuente) 28.000 descriptores. Teniendo en cuenta que cada resumen tiene una media de 100 palabras y en el prototipo se trabaja con más de 870.000 registros, se trata de un procedimiento costoso. El tiempo de procesado en la prueba bajó de 1,6h a sólo 30 minutos. Sin embargo, se decidió mantener el proceso pues se cree que es importante para el proyecto. Se asume que si la palabras clave resultantes del preprocesado se encuentran en un tesauro será más sencillo para el recomendador encontrar coincidencias.

El resultado de este preprocesado se almacena en una nueva família en la tabla "Documents" de HBase. Concretamennte de guarda una lista con diez palabras clave, su frecuencia de aparición y un booleano que indica si la palabra se encontraba en el tesauro (ver figura 12.

```
hbase(main):001:0> scan 'Documents', {COLUMNS => 'Keywords'}
ROW                           COLUMN+CELL
 http://hdl.handle.net/10803/100 column=Keywords:col1, timestamp=1441580074590, value=[[protein, 2, true], [dna, 1, true], [bot
 0                              ryti, 2, true], [gene, 4, true], [rna, 1, true], [bacteria, 1, true], [aspergillu, 2, true], [
                                transgen, 3, true], [magnaporth, 2, true], [planta, 13, false]]
 http://hdl.handle.net/10803/100 column=Keywords:col1, timestamp=1441580074589, value=[[softwar, 1, true], [factor, 1, true], [
 00                             informacion, 19, false], [trafico, 13, false], [servicio, 8, false], [web, 6, false], [usuario
                                , 5, false], [ontologia, 5, false], [vial, 5, false], [sistema, 4, false]]
 http://hdl.handle.net/10803/100 column=Keywords:col1, timestamp=1441580074903, value=[[individu, 1, true], [negociacion, 16, f
 01                             alse], [sistema, 8, false], [protocolo, 8, false], [elemento, 7, false], [acuerdo, 6, false],
                                [participant, 5, false], [entorno, 4, false], [problema, 4, false], [informatico, 3, false]]
 http://hdl.handle.net/10803/100 column=Keywords:col1, timestamp=1441580075341, value=[[molt, 1, true], [social, 11, true], [el
 02                             ement, 1, true], [environ, 1, true], [abstract, 1, true], [semant, 1, true], [intellig, 1, tru
                                e], [del, 12, false], [el, 11, false], [i, 8, false]]
```

Figure 12: Datos preprocesados de un artículo en HBase

10.3 Recomendación

Se ha intentado usar el algoritmo DIMSUM para el proceso de recomendación. Sin embargo, no se ha tenido éxito ya que la entrada del algoritmo funciona con tipos numéricos (float) en vez de los Strings (*keywords*) requeridos en nuestra aproximación. En los ejemplos [13] no se ha encontrado ningún caso donde poder definir una RowMatrix con Strings, y no se ha dispuesto de tiempo suficiente para realizar una adaptación.

En este punto, y como era importante que el prototipo pudiera realizar alguna recomendación y esta fuera distribuida a través del siguiente módulo, se ha decidido implementar un sencillo algoritmo (ver 3) de fuerza bruta contra un subconjunto más reducido de los artículos recolectados.

El resultado del algoritmo es una matriz donde cada fila esta formada por parejar identificador-puntuación que representan la distancia con el registro correspondiente al índice de aquella fila. un ejemplo es:

Algoritmo 3 Cálculo de distancia entre artículos (fuerza bruta)

Entrada: Listado de keywords de artículos en HBase
Salida: Matriz de pares identificador, puntuación

```
1: Obtener los registros de keywwords de HBase
2: para todos los registros de keywords (reg1) hacer
3:    para todos los registros de keywords (reg2) hacer
4:       si Reg2 contiene la keyword actual entonces
5:          puntuación + 1
6:       fin si
7:       guardamos reg2 y su puntuación
8:    fin paraGuardamos la lista de puntuaciones de reg1 en la matriz final
9: fin para
```

```
[[[http://...net/1000, 10],[http://...net/1001, 5],[http://...net/1002, 0], ...],
[[http://...net/1000, 0],[http://...net/1001, 10],[http://...net/1002, 4], ...]]
```

Esta información permite, una vez computadas todas las distancias, ordenar cada fila de la matriz por puntuaciones y quedarse con las N primeras puntuaciones (se han escogido 5 para esta prueba de concepto). Esto da un resultado de parejas clave-valor del siguiente estilo:

```
http://hdl.handle.net/10803/1000: [http://hdl.handle.net/10803/1000,
http://hdl.handle.net/10803/1001, http://hdl.handle.net/10803/10045,
http://hdl.handle.net/10803/10068, http://hdl.handle.net/10803/10512]
```

Este formato se adapta perfectamente a la manera como se desea guardar la lista de recomendados, un objeto JSON que permitirá acceder, dada una URL de un artículo, al listado de sus artículos recomendados.

Como era previsible, la solución implementada por fuerza bruta no es admisible desde el punto de vista de eficiencia computacional por su elevado número de iteraciones, por lo que se ha reducido a diez los artículos de los que se han calculado todas las distancias y se han almacenado posteriormente en Couchbase.

A continuación se muestra el tiempo de computación de cada uno de estos diez artículos y seguidamente los resultados, es decir, un listado con los diez artículos con sus cinco artículos recomendados:

```
1/870194 done Tiempo de ejecución: 13.994s
2/870194 done Tiempo de ejecución: 14.714s
3/870194 done Tiempo de ejecución: 14.64s
4/870194 done Tiempo de ejecución: 15.526s
5/870194 done Tiempo de ejecución: 76.96s
6/870194 done Tiempo de ejecución: 86.808s
7/870194 done Tiempo de ejecución: 14.152s
8/870194 done Tiempo de ejecución: 28.02s
```

```
9/870194 done Tiempo de ejecución: 66.884s
10/870194 done Tiempo de ejecución: 46.792s

ITEM0: http://hdl.handle.net/10803/1000
recomendado1: http://hdl.handle.net/10803/1000 con puntuacion 10
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC93851/ con puntuacion 5
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC92586/ con puntuacion 5
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC514576/ con puntuacion 5
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC353920/ con puntuacion 5
ITEM1: http://hdl.handle.net/10803/10000
recomendado1: http://hdl.handle.net/10803/10000 con puntuacion 10
recomendado1: http://hdl.handle.net/10803/96331 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/9621 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/9187 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/8805 con puntuacion 3
ITEM2: http://hdl.handle.net/10803/10001
recomendado1: http://hdl.handle.net/10803/10001 con puntuacion 10
recomendado1: http://hdl.handle.net/10803/10004 con puntuacion 4
recomendado1: http://hdl.handle.net/10803/6874 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/41718 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/292369 con puntuacion 3
ITEM3: http://hdl.handle.net/10803/10002
recomendado1: http://hdl.handle.net/10803/10002 con puntuacion 10
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1008277/ con puntuacion 6
recomendado1: http://hdl.handle.net/10803/9122 con puntuacion 6
recomendado1: http://hdl.handle.net/10803/51341 con puntuacion 6
recomendado1: http://hdl.handle.net/10803/5090 con puntuacion 6
ITEM4: http://hdl.handle.net/10803/10003
recomendado1: http://hdl.handle.net/10803/10003 con puntuacion 9
recomendado1: http://hdl.handle.net/10803/10004 con puntuacion 5
recomendado1: http://hdl.handle.net/10803/1595 con puntuacion 4
recomendado1: http://hdl.handle.net/10803/9999 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/9992 con puntuacion 3
ITEM5: http://hdl.handle.net/10803/10004
recomendado1: http://hdl.handle.net/10803/10004 con puntuacion 9
recomendado1: http://hdl.handle.net/10803/10003 con puntuacion 5
recomendado1: http://hdl.handle.net/10803/56317 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/1595 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/9999 con puntuacion 2
ITEM6: http://hdl.handle.net/10803/10005
recomendado1: http://hdl.handle.net/10803/10005 con puntuacion 10
recomendado1: http://hdl.handle.net/10803/9668 con puntuacion 4
recomendado1: http://hdl.handle.net/10803/4771 con puntuacion 4
recomendado1: http://hdl.handle.net/10803/285751 con puntuacion 4
recomendado1: http://hdl.handle.net/10803/10095 con puntuacion 4
ITEM7: http://hdl.handle.net/10803/10006
recomendado1: http://hdl.handle.net/10803/10006 con puntuacion 10
recomendado1: http://hdl.handle.net/10803/9843 con puntuacion 6
```

```
recomendado1: http://hdl.handle.net/10803/9978 con puntuacion 5
recomendado1: http://hdl.handle.net/10803/96723 con puntuacion 5
recomendado1: http://hdl.handle.net/10803/8412 con puntuacion 5
ITEM8: http://hdl.handle.net/10803/10007
recomendado1: http://hdl.handle.net/10803/10007 con puntuacion 10
recomendado1: http://hdl.handle.net/10803/7325 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/6790 con puntuacion 3
recomendado1: http://hdl.handle.net/10803/117618 con puntuacion 3
recomendado1: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC58814/ con puntuacion 2
ITEM9: http://hdl.handle.net/10803/10008
recomendado1: http://hdl.handle.net/10803/10008 con puntuacion 5
recomendado1: http://hdl.handle.net/10803/8553 con puntuacion 2
recomendado1: http://hdl.handle.net/10803/8475 con puntuacion 2
recomendado1: http://hdl.handle.net/10803/83983 con puntuacion 2
recomendado1: http://hdl.handle.net/10803/8340 con puntuacion 2
```

Se ha decidido no excluir a los propios artículos del proceso de recomendación para comprobar, con éxito, que la herramienta detecta esta gran similitud (del 100%).

Por otro lado, se ha comprobado que la idea inicial de **crear un recomendador de artículos de repositorios digitales diferentes es viable**. Como se aprecia en el listado de resultados anterior, a un artículo del repositorio de tesis "*Aplicaciones biotecnológicas del gen 'afp' (Antifungical Protein) de 'Aspergillus giganteus' para la protección de plantas frente a infección por patógenos*" (ver figura 13) se le están recomendando artículos de **otro** repositorio digital, Pubmed, como "*Bacterioferritin A Modulates Catalase A (KatA) Activity and Resistance to Hydrogen Peroxide in Pseudomonas aeruginosa*" (ver figura 14). Se ha comprobado que el artículo de Pubmed no se encuentra citado en las referencias de la tesi doctoral. Este podría ser un buen ejemplo de herramienta de descubierta de conocimiento.

10.4 Distribución de los datos

Se ha instalado un Couchbase Server en modo local, creando un *bucket* de datos especifico (ver figura 15. Como key se usa el identificador del artículo (su URL unívoca) del que se quieren almacenar sus artículos recomendados.

Para poder acceder a los recomendados de un artículo en concreto se ha creado una vista para este *bucket*, en donde en el map emitimos el id del artículo y la lista de recomendados como valor.

De este modo, al ejecutar la siguiente consulta poniendo por clave un artículo, se obtiene la lista de sus artículos recomendados (ver figura 17):

`http://localhost:8092/recomendador/_design/dev_recomendador/_view/recomendador?stale=false&inclusive_end=true&connection_timeout=60000&key=%22http://hdl.handle.net/10803/10000%22`

Este es el mecanismo de integración que se usaría desde los repositorios digitales para acceder a las URLs de los artículos relacionados.

Figure 13: Artículo en TDX

Figure 14: Artículo en PubMed

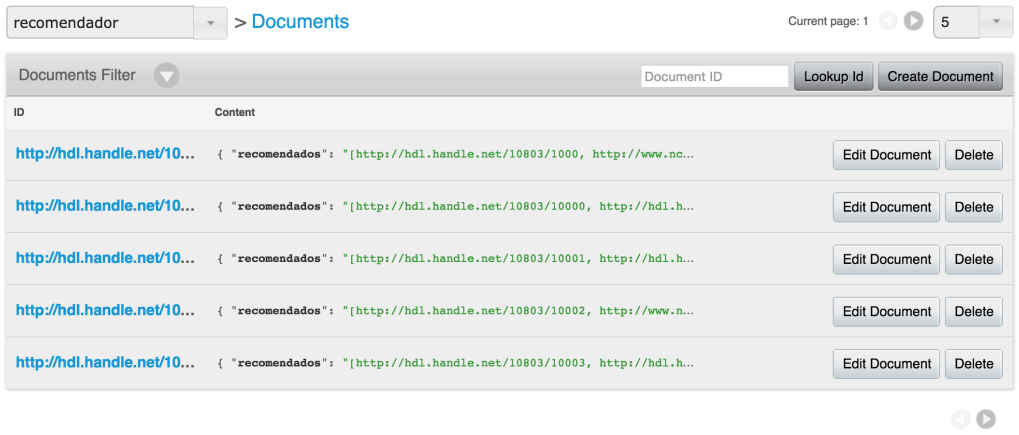


Figure 15: Bucket de datos en Couchbase

Figure 16: Artículo en Couchbase y las URLs de sus cinco recomendados

Figure 17: Artículo consultado vía API y las URLs de sus cinco recomendados

En el ejemplo, los artículos recomendados del artículo `http://hdl.handle.net/10803/1000` son:

- recomendado1: `http://hdl.handle.net/10803/1000` con puntuacion 10
- recomendado2: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC93851` con puntuacion 5
- recomendado3: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC92586` con puntuacion 5
- recomendado4: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC514576` con puntuacion 5
- recomendado5: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC353920` con puntuacion 5

11 Conclusiones y trabajo futuro

El objetivo del proyecto era **realizar un recomendador de artículos científicos**. Se ha conseguido desarrollar un prototipo funcional de recomendador de contenido basado en palabras clave extraídas de metadatos de dos repositorios digitales, TDX y Pubmed.

Se han recolectado metadatos de 850K artículos y se han almacenado en una base de datos HBase. Se han preprocesado estos metadatos y para cada artículo se han obtenido 10 palabras clave. Para una muestra reducida de artículos se han calculado sus puntuaciones de similitud con el resto de artículos y, para cada artículo, se ha almacenado en una caché de vistas sus cinco artículos más similares (recomendados). Esta caché puede ser accedida vía API desde repositorios digitales, potenciales clientes del producto mediante un modelo de suscripción.

Como se ha descrito en el apartado de trabajos previos, los recomendadores que actualmente funcionan en repositorios digitales sólo recomiendan contenidos de los propios autores o del propio repositorio. La vocación de este proyecto ha sido conseguir un recomendador "global", que recomiende tanto artículos internos como externos al propio repositorio, basado en metadatos de "todos" los repositorios existentes. En el prototipo, con sólo dos repositorios, **se ha conseguido validar la viabilidad del objetivo al comprobar como a artículos del repositorio de tesis TDX se le recomendavan otros "similares" del repositorio Pubmed**.

Por otro lado, un objetivo paralelo del prototipo era la obtención de métricas de rendimiento que permitan realizar estimaciones del sistema completo y analizar su escalabilidad y viabilidad. Esto se ha cumplido de manera parcial. Diversos imprevistos y problemas en la recolección y el preprocesado de los datos explicados en apartados anteriores no han permitido llegar a una solución completa y aceptable, en términos de rendimiento, del módulo de recomendación. Al trabajar con una muestra reducida no ha sido posible completar todos

los indicadores necesarios (de volumen y velocidad) para asegurar que la escalabilidad del proyecto en este punto es viable.

Se intuye que aumentando los recursos computacionales y adaptando el algoritmo DIMSU u otro de similar en el que no sea necesario el empleo de la fuerza bruta el proyecto será viable técnicamente. Más difícil de comprobar será la sostenibilidad económica, es decir, los ingresos que el modelo de negocio deberá generar para cubrir los costes de este augmento de infraestructura. Esta segunda prueba de concepto podrá ser efectuada el la siguiente iteración de la herramienta.

A medida que se ha avanzado en el proyecto han surgido nuevas ideas o diferentes alternativas a validar. Para mantener el foco en el objetivo inicial y conseguir resultados en el (muy corto) periodo de tiempo marcado del proyecto, a continuación se detallan las diferentes alternativas que se podrían explorar en futuras iteraciones.

- Herramienta para la detección de artículos duplicados en diferentes repositorios. Como se ha comprobado en los resultados, el propio artículo era el recomendado con mayor puntuación. Si se produce una puntuación máxima pero el artículo es de otro repositorio se detecta el duplicado.
- Incluir la componente temporal de los artículos en el recomendador.
- Basar el recomendador en el texto completo de los própios artículos, en lugar de en sus metadatos. Su recolección es posible a través del protocolo OAI-ORE[28].

Agradecimientos

Se agradece a los tres tutores de este proyecto, Óscar Romero, Víctor Herrero y Pere Torrents, su ayuda y soporte para la realización de esta primera versión del recomendador de *outputs* científicos. También valoramos las aportaciones de Lluís Belanche y Tomás Aluja y extendemos el agradecimiento al resto de profesorado del primer postgrado en Big Data Management and Analitycs de la Universitat Politècnica de Catalunya (UPC-BarcelonaTech)

Referencias

[1] Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor *Recommender Systems Handbook*, Springer, 2011.

[2] Linyuan Lü, Matús Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang *Recommender Systems*, Physics Reports 519, pp 1-59, 2012.

[3] Greg Linden, Brent Smith, Jeremy York *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, IEEE Internet Computing archive. Volume 7 Issue 1, pp 76-80. January 2003

[28] `https://www.openarchives.org/ore`

[4] Xavier Amatriain, Justin Basilico *Netflix Recommendations: Beyond the 5 stars*, http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html April 6, 2012.

[5] Lukás Brozovský, Václav Petrícek *Recommender System for Online Dating Service*, arXiv:cs/0703042v1, Mars, 9, 2007.

[6] Sung-Hwan Min, Ingoo Han *Detection of the customer time-variant pattern for improving recommender systems*, Expert Systems with Applications 28 pp. 189-199, 2005.

[7] Brittany Darwell *Graph Search: a recommendation engine only Facebook could power*, http://www.adweek.com/socialtimes/graph-search-a-recommendation-engine-only-facebook-could-power/289173 January 16, 2013.

[8] Lili Wu *Browsemap: Collaborative Filtering At LinkedIn*, https://engineering.linkedin.com/recommender-systems/browsemap-collaborative-filtering-linkedin October 23, 2014.

[9] Sander Dieleman *Recommending music on Spotify with deep learning*, http://benanne.github.io/2014/08/05/spotify-cnns.html August, 2014.

[10] Daniel Valcarce, Javier Parapar, Álvaro Barreiro *When Recommenders Met Big Data: An Architectural Proposal and Evaluation*, 3rd Spanish Conference on Information Retrieval June 19-20, 2014

[11] Reza Zadeh *All-pairs similarity via DIMSUM*, August 29, 2015 https://blog.twitter.com/2014/all-pairs-similarity-via-dimsum, [consultado el 5-09-15]

[12] Reza Zadeh *Distributing Matrix Computations with Spark MLlib*, 2015 https://blog.twitter.com/2014/all-pairs-similarity-via-dimsum, http://stanford.edu/~rezab/slides/reza_mllib_maryland.pdf [consultado el 5-09-15]

[13] Jonathan Alter *CosineSimilarity.scala*, 2015. https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/mllib/CosineSimilarity.scala [consultado el 5-09-15]

[14] Nitin Agarwal, Ehtesham Haque, Huan Liu, Lance Parsons *Research Paper Recomender Systems: A Subspace Clustering Approach*, 6th International Conference, WAIM 2005, Hangzhou, China, October 11-13, 2005.

[15] Zhiping Zhang, Linna Li *A research paper recommender system based on spreading activation model*, 2nd International Conference on Information Science and Engineering (ICISE), pp.928-931, December 4-6, 2010.

[16] Joonseok Lee, Kisung Lee, Jennifer G. Kim *Personalized Academic Research Paper Recommendation System*, eprint arXiv:1304.5457 April, 19, 2013.

[17] Ming Zhang, Weichun Wang, Xiaoming Li *A Paper Recommender for Scientific Literatures Based on Semantic Concept Similarity*, LNCS 5362, pp. 359-362 2008.

[18] Zan Huang, Wingyan Chung, Thian-Huat Ong, Hsinchun Chen *A graph-based recommender system for digital library*, Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries pp. 65-73 ACM New York, 2002.

[19] Ajith Kodakateri, Susan Gauch, Hiep Luong, Joshua Eno *Conceptual Recommender System for CiteSeer*, RecSys'00, October, 23-25, 2009.

[20] Shu-Chuan Liao, Kuo-Fong Kao, I-En Liao, Hui-Lin Chen, Shu-O Huang, *PORE: a personal ontology recommender system for digital libraries*, The Electronic Library, Vol. 27 Iss: 3, pp.496 - 508, 2009.

[21] Roberto Torres, Sean M. McNee, Mara Abel, Joseph A. Konstan, John Riedl *Enhancing Digital Libraries with TechLens*, JCDL'04, November 2004.

[22] Bela Gipp, Jöran Bell, Christian Hentschel *Scienstein: A Research Paper Recommender System*, Proceedings of the International Conference on Emerging Trends in Computing (ICETiC'09), pp. 309-315, Virudhunagar, India January, 2009.

[23] Robin Burke *Hybrid Recommender Systems: Survey and Experiments*, User Modeling and User-Adapted Interaction, Volume 12, Issue 4, pp 331-370, November, 2002.

[24] Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, John Riedl *On the Recommending of Citations for Research Papers*, CSCW'02, November, 16-20, 2002.

[25] Sean M. McNee, Nishikant Kapoor, Joseph A. Konstan *Don't Look Stupid: Avoiding Pitfalls when Recomending Research Papers*, CSCW'06, Banff, Alberta, Canada November, 4-8, 2006.

[26] Desmond Elliot, James Rutherford, John S. erickson *A Recommender System for the DSpace Open Repository Platform*, Hewlett-Packard Laboratories, Bristol, April, 7 2008.

[27] A. Ruiz-Iniesta, G. Jiménez-Díaz, M. Gómez-Albarrán *Personalización en Recomendadores Basados en Contenido y su Aplicación a Repositorios de Objetos de Aprendizaje*, IEEE-RITA, Vol. 5, Núm 1, February, 2010.

[28] Toine Bogers, Antal van den Bosch *Recomending Scientific Articles Using CiteULike*, RecSys'08, October, 23-25, 2008.

[29] Woracit Choochaiwattana *Usage of Tagging for Research Paper Recomendation*, 3rd International Conference on Advanced Computer Theory and Enginyeering, 2010.

[30] Ricard de la Vega *Interoperatibilitat entre repositoris de material docent i recol·lectors de metadades*, UOC. 2012. `http://hdl.handle.net/10609/15761` [consultado el 5-09-15]

[31] Peter Willet *The Porter stemming algorithm: then and now*, Electronic Library and Information Systems, 40 (3) pp.219-223 2006.

[32] *Listado de Stop Words en castellano*, `https://code.google.com/p/stop-words/source/browse/trunk/stop-words/stop-words-spanish.txt?r=3`

[33] Oscar Romero *Document Stores*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[34] Alberto Abelló *Key-value and Wide-column stores*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[35] Anna Queralt *Open Data: Why and How*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[36] Sergi Nadal *Mining Big Datasets With Spark MLlib*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[37] Sergi Nadal *Apache Spark*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[38] Víctor Herrero *Introduction to Hadoop*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[39] Víctor Herrero *Setting-up of a HDFS cluster*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[40] Víctor Herrero *Importing and compiling with Eclipse*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[41] *SparkOnHBase*, Librería para facilitar la conexión Spark-HBase `https://github.com/tmalaska/SparkOnHBase` [consultado el 5-09-15]

[42] *HBase-shell-commands* `https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands` [consultado el 5-09-15]

[43] Mohammad Farooq *Loading data in HBase Tables on HDInsight using bult-in ImportTsv utility* `http://blogs.msdn.com/b/bigdatasupport/archive/2014/12/12/loading-data-in-hbase-tables-on-hdinsight-using-bult-in-importtsv-utility.aspx` [consultado el 5-09-15]

[44] William Benton *Improving Spark application performance.* September 9, 2014 `http://chapeau.freevariable.com/2014/09/improving-spark-application-performance.html` [consultado el 5-09-15]

[45] *ZooKeeper Getting Started Guide* `http://zookeeper.apache.org/doc/r3.1.2/zookeeperStarted.html` [consultado el 5-09-15]

[46] Enis Soztutar *Apache HBase Region Splitting and Merging*. February 1, 2013 `http://hortonworks.com/blog/apache-hbase-region-splitting-and-merging/` [consultado el 5-09-15]

[47] *Configuring HBase in Pseudo-distributed Mode*. `http://www.cloudera.com/content/cloudera/en/documentation/cdh4/v4-2-1/CDH4-Installation-Guide/cdh4ig_topic_20_5.html` [consultado el 5-09-15]

[48] Adrià Batlle *Business Modelling*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[49] Alexander Osterwalder, Yves Pigneur, Alan Smith et altri *Business Model Generation*, Wiley. 2010.

[50] Eric Ries *The lean startup: how today's entrepreneurs use continuous innovation to create radically successful businesses*, Crown Publishing. 2011.

[51] Pere Torrents *Financiación de proyectos y marketing*, Material docente del I postgrado en Big Data Management and Analitycs. UPC-Barcelonatech. 2015.

[52] Philipp Max Hartmann, Mohamed Zaki, Niels Feldmann, Andy Neely *Big Data for Big Business? A Taxonomy of Data-driven Business Models used by Start-up Firms*, Cambridge Service Alliance, March, 2014.

A Estructura Dublin Core de un artículo científico

Ejemplo de artículo científico del repositorio PubMed Central:

```
<record>

  <header>
        <identifier>oai:pubmedcentral.nih.gov:13900</identifier>
        <datestamp>2001-02-27</datestamp>
        <setSpec>brcnres</setSpec>
        <setSpec>pmc-open</setSpec>
  </header>

  <metadata>
        <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
 xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xsi="http://www.w3.org/
␣2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org
␣/OAI/2.0/oai_dc/␣http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
        <dc:title>
                Comparison of written reports of mammography, sonography and
                 magnetic resonance mammography for preoperative evaluation of
                 breast lesions, with special emphasis on magnetic resonance
                 mammography
```

```
        </dc:title>
        <dc:creator>Malur, Sabine</dc:creator>
        <dc:creator>Wurdinger, Susanne</dc:creator>
        <dc:creator>Moritz, Andreas</dc:creator>
        <dc:creator>Michels, Wolfgang</dc:creator>
        <dc:creator>Schneider, Achim</dc:creator>
        <dc:subject>Primary Research</dc:subject>
        <dc:publisher>BioMed Central</dc:publisher>
        <dc:date>2001</dc:date>
        <dc:date>2000-11-02</dc:date>
        <dc:identifier>
                http://www.ncbi.nlm.nih.gov/pmc/articles/PMC13900/
        </dc:identifier>
        <dc:identifier>
                http://www.ncbi.nlm.nih.gov/pubmed/11250746
        </dc:identifier>
        <dc:description>
                Patients with abnormal breast findings (n = 413) were
                examined by mammography, sonography and magnetic resonance
                (MR) mammography; 185 invasive cancers, 38 carcinoma in situ
                and 254 benign tumours were confirmed histologically.
                Sensitivity for mammography was 83.7%, for sonography it was
                89.1% and for MR mammography it was 94.6% for invasive cancers.
                In 42 patients with multifocal invasive cancers, multifocality
                had been detected by    mammography and sonography in 26.2%, and
                by MR mammography in 66.7%. In nine patients with multicentric
                cancers, detection rates were 55.5, 55.5 and 88.8%,
                respectively. Carcinoma in situ was diagnosed by mammography
                in 78.9% and by MR mammography in 68.4% of patients.
                Combination of all three diagnostic methods lead to the best
                results for detection of invasive cancer and multifocal
                disease. However, sensitivity of mammography and sonography
                combined was identical to that of MR mammography (ie 94.6%).
        </dc:description>
        <dc:type>Text</dc:type>
        <dc:language>en</dc:language>
        <dc:rights>
                Copyright  2000 BioMed Central Ltd on behalf of the copyright
                holders
        </dc:rights>
        </oai_dc:dc>
  </metadata>
</record>
```

B Análisis de costes

Al usar *software* no comercial, como se detalla en la figura 5.1, los costes del proyecto serán los relativos a personal e infraestructura de *cloud*.

En cuanto a personal, el primer año se estima que será necesario la participación de un *data scientist* a tiempo completo, una tercera parte de un administrador de sistemas y media persona para la coordinación y comercialización de la solución. En total, aproximadamente 65K euros.

Respecto la infraestructura, como se comenta en las conclusiones, no se dispone de suficientes indicadores como para dimensionar correctamente el entorno. De todos modos, se realiza la siguiente estimación:

```
SERVER_NAME: EC2_type vCPU RAM $precio)

hadoop-nn: c4.xlarge 4 7.5 $0.22 por hora
hadoop-dn1: m4.large 2 8 $0.126 por hora
hadoop-dn2: m4.large 2 8 $0.126 por hora
couchbase: t2.large 2 8 $0.104 por hora
couchbase-01: c4.xlarge 4 7.5 $0.22 por hora
couchbase-02: c4.xlarge 4 7.5 $0.22 por hora

TOTAL $1,02/hora = $8.900/año

El mismo ejercicio con instancias reservadas 1 año:
hadoop-nn: c4.xlarge $0.1384 ($1212 anual)
hadoop-dn1: m4.large $0.0725  ($635 anual)
hadoop-dn2: m4.large $0.0725  ($635 anual)
couchbase: t2.large $0.0689  ($604 anual)
couchbase-01: c4.xlarge $0.1384 ($1212 anual)
couchbase-02: c4.xlarge $0.1384 ($1212 anual)

TOTAL $5.510/año

El mismo ejercicio con instancias reservadas 3 años:
hadoop-nn: c4.xlarge $0.0859 ($753 anual = $2258 total)
hadoop-dn1: m4.large $0.0047 ($412 anual = $1235 total)
hadoop-dn2: m4.large $0.0047 ($412 anual = $1235 total)
couchbase: t2.large $0.0462 ($405 anual = $1214 total)
couchbase-01: c4.xlarge $0.0859 ($753 anual = $2258 total)
couchbase-02: c4.xlarge $0.0859 ($753 anual = $2258 total)

TOTAL $3.488/año con compromiso a 3 años
```

En cuanto al tráfico:

- Trafico entre máquinas: no facturado
- Tráfico entrante (recolección de datos): no facturado
- Eestimación de tráfico saliente a internet menor a 1 GB/mensual (AVG 34 MB/día).

Si consideramos la reserva de un año de la infraestructura, los costes aproximados del proyecto son:

- Personal: 65K euros (5,5K euros/mes)
- Infraestructura: 5.5K dolars = 4.9K euros (400 euros/mes)
- Contingencia: 15% = (65K+4.9K)*0.15 = 10.5K (875 euros/mes)
- **Total: 81K euros (6.750 euros/mes[29])**

[29]Por simplificación, se considera que tanto el personal como los costes del *cloud* se disponen desde el primer mes, y así es posible distribuirlos equitativamente a lo largo de los doce meses del año.