UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Estudio de aplicabilidad de técnicas de Big Data en el streaming multimedia para la detección de eventos

Departamento de Sistemas Informáticos y Computación
Master Universitario en Computación Paralela y Distribuida
Universitat Politècnica de València
Curso Académico 2014 - 15

Autor
José Herrera Hernández

Director
Germán Moltó Martínez

Valencia, Febrero de 2015

An EXPERT is a person who has made all the mistakes
that can be made in a narrow field...
--- Niels Bohr ---

A MASTER is one who has LEARNED from those mistakes...
-- Jay Stoddard --

Abstract

The massive amount of multimedia information currently available through the Internet demands efficient techniques to extract knowledge from that Big Data. In this work, we propose an architecture to capture, process, analyze and visualize data coming from multiple streaming multimedia TV stations and radio stations. For that, we rely on the Hadoop framework available within the IBM InfoSphere BigInsights platform. We create a workflow to automate the different stages that range from Automatic Speech Recognition using open-source tools to visualization by means of the R framework. We emphasize techniques such as diarization and the optimization of the number of Hadoop nodes, provisioned from Cloud infrastructures, to deliver enhanced performance.

The results show that it is possible to automate knowledge extraction from multimedia data running on virtualized infrastructure by means of Big Data techniques.

Keywords: Big Data, Hadoop, MapReduce, multimedia, R, BigInsights

Resumen

La gran cantidad de información multimedia actualmente disponible en Internet demanda técnicas eficientes para la extraer conocimiento de los grandes volúmenes de datos. En este trabajo, se propone una arquitectura para capturar, procesar, analizar y visulizar información proveniente de streaming multimedia como emisoras de televisión o radio. Para ello, se utiliza la infraestructura de Hadoop existente en la plataforma IBM InfoSphere BigInsights. Se ha creado un flujo de trabajo para automatizar las diferentes etapas de las que se compone, extendiendose desde el reconocimiento vocal utilizando herramientas de código abierto hasta la visualización utilizando R. Se utilizan técnicas como la diarización y la optimización del número de nodos en Hadoop, que son aprovisionados por medio de infraestructuras Cloud para su mejor desempeño.

Los resultados muestran que es posible automatizar la extracción del conocimiento de datos multimedia utilizando infraestructuras virtuales por medio de técnicas de Big Data.

Palabras clave: Big Data, Hadoop, MapReduce, multimedia, R, BigInsights

Reconocimientos

Quisiera, como no, reconocer a unas cuantas personas la realización de este trabajo que supone la finalización de un periodo de trabajo muy importante pero también el inicio de un ciclo que imagino que será fructífero. Primero agradecer, por supuesto, al director de mi proyecto, con el que las reuniones han sido productivas y enriquecedoras en todo momento, guiándome en la selección de las metas y mostrándome aquellos problemas ocultos en la concepción del proyecto. Quisiera agradecer también al GRyCAP (Grupo de Grid y Computación de Altas Prestación) de la Universitat Politècnica de València por la cesión de la infraestructura para la ejecución de los nodos necesarios para las pruebas descritas en este documento. Y por último a todos los profesores del Master Universitario en Computación Paralela y Distribuida de la UPV por haber fomentado y creado un interés en este rama tan interesante de la computación.

Por otro lado quisiera agradecer a toda mi familia el interés por que pudiera realizar este proyecto. A mi mujer Beatriz por aguantar tantas horas de paseo con los niños mientras me dedicaba a completar tareas y a Pablo y Ana que saben que su padre siempre está *"!!jugando con el ordenador* :-)¡¡".

Tabla de Contenidos

Lista de Figuras

Lista de Formulas

Lista de Tablas

1 Introducción

Debido al aumento de la capacidad de transmisión de datos en las redes de comunicación y el incremento de la potencia de procesamiento de los dispositivos, el contenido multimedia se ha democratizado y extendido a una amplia variedad de dispositivos.

Debido al incremento exponencial de documentos con contenido audiovisual en Internet, en este trabajo se desea plantear la necesidad de su tratamiento y la obtención de información útil para su consumo. Según el informe sobre los contenidos digitales en España [46] de la ONTSI (Observatorio Nacional de las Telecomunicaciones y de la Sociedad de la Información) se prevé un aumento considerable de los contenidos multimedia, en segundo puesto, justo por detrás de los contenidos procedentes de los juegos online. Las previsiones de CISCO [10] sobre el aumento del tráfico multimedia en dispositivos móviles en este sentido son abrumadoras. Un aumento del 60% en contenidos de vídeo por Internet pone a la cabeza este tipo de datos e incrementa la necesidad de '*entender*' esta información.

La obtención de datos mediante el tratamiento de información con base textual se incrementa constantemente, pero el tratamiento de contenidos multimedia, por su propia naturaleza, no ha evolucionado con la misma velocidad. Incluso las herramientas más utilizadas hacen especial hincapié en la utilización de textos para la obtención de información utilizable por terceros. El volumen de datos en TeraBytes, que no en número de documentos, incluso es mayor en el caso de ficheros audiovisuales que en los textuales.

En este punto, se desea plantear la posibilidad de tratar el audio, proporcionado de forma autónoma o asociado a vídeos, para la obtención de información. En nuestro caso se centrará única y exclusivamente en la utilización de transmisiones de audio/vídeo de medios televisivos y de radio, todos ellos con emisiones en Internet, conocido comúnmente como *streaming*.

La utilización de contenidos no textuales en su origen plantea una dificultad añadida en su tratamiento. Es necesaria una conversión adecuada del contenido multimedia a un sistema que permita la búsqueda de patrones o el hallazgo de singularidades. La escalabilidad también representa otra de las dificultades de estos sistemas basados en audio/vídeo. Si la información multimedia se populariza a los dispositivos móviles, como parece que es la tendencia actual, los sistemas encargados de este tratamiento deben ser capaces de adaptarse a esas necesidades y al volumen de datos que esto supone.

Intuitivamente parece necesaria la utilización de sistemas que conjuguen la capacidad de tratamiento de ficheros con datos no estructurados y que al mismo tiempo sean capaces de almacenar grandes volúmenes de información. En este sentido se ha acuñado el término Big Data y se han desarrollado sistemas que cumplen con estas necesidades de procesamiento y almacenamiento.

Aunque los medios de difusión tradicionales son más "perezosos" en proporcionar información sobre eventos puntuales y teniendo en cuenta que las redes sociales son más dinámicas en este aspecto [23] [36], existen cuestiones que debemos tener en cuenta como son la veracidad de los medios utilizados, la capacidad para organizar de contenidos o su disponibilidad por parte de los usuarios a lo largo del día.

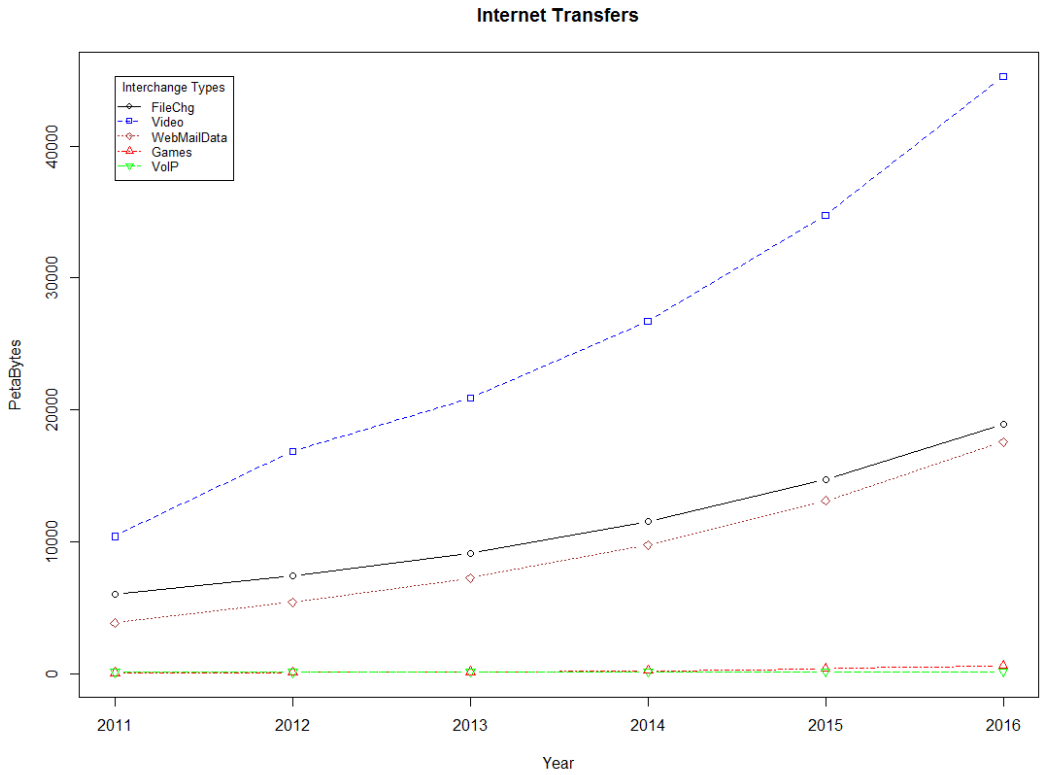

Figura 1 - Tráfico usuarios Internet (2011-2016) PetaBytes_Mes
Fuente: Cisco VNI, 2014 [10]

El tratamiento del vídeo está muy por debajo de los tratamientos que se realizan en los medios sociales de los datos textuales. Por ejemplo, son pocos los ejemplos que aparecen del tratamiento de los vídeos de plataformas como YouTube para obtener datos del comportamiento de las personas (de qué tratan los últimos vídeos que se han subido ó cuales son los temas de interés de los últimos vídeos visualizados). Son casos donde la información de partida es el vídeo y por tanto denota que existe mucha información pendiente de procesar.

	2011	2012	2013	2014	2015	2016
Intercambio de archivos	6.013	7.403	9.153	11.569	14.758	18.892
Vídeo por Internet	10.423	16.880	20.904	26.722	34.755	45.280
Internet, email y datos	3.863	5.422	7.274	9.783	13.119	17.583
Juegos online	77	115	170	251	404	630
Voz sobre datos (VoIP)	147	154	159	163	169	174
Total	20.523	29.974	37.660	48.488	63.204	82.560

Tabla 1 - Tráfico usuarios Internet (2011-2016) PetaBytes-Mes.
Fuente Cisco VNI, 2014 [10]

Lo que sí parece evidente es la necesidad de desarrollar métodos que permitan la obtención y el tratamiento de información de estas emisiones que tanta aceptación tienen por parte de la sociedad. Este proyecto se ha diseñado con vistas a poder verificar el modelo de tratamiento MapReduce en los procesos de detección de eventos, esto es, comprobar que los sistemas diseñados son capaces de diferenciar, entre la multitud de eventos que se puede emitir, aquellos que resultan importantes por su amplia difusión en estos medios o por alguna otra razón.

En el contexto regional de este trabajo (Comunidad Valenciana, España) podemos beneficiarnos del conocimiento en las áreas del turismo como se hace en [49] lo que puede tener gran repercusión en un sector que en esta región tiene gran desarrollo. Esto permitiría detectar ciertos eventos en los contenidos que se estudien para llegar a obtener conclusiones que pueden mejorar nuestra gestión del mundo de los servicios. Por ejemplo, la existencia de numerosas menciones,

en TV o radio, a conflictos sociales asociados a países del Norte de África puede suponer una mejora de las expectativas en las pernoctaciones de extranjeros en la época estival.

Existe una evaluación en [47] de las tareas realizadas para ASR (Automatic Speech Recognition) en una cantidad pequeña de procesadores. El tamaño de su prueba llega a un par de procesadores y con máximo de 8 cores. Esto nos lleva a la posibilidad de estudiar la utilización de una plataforma de cómputo distribuido con el fin de repartir la carga de trabajo. Una de la posibilidades, y la que tiene más aceptación, es Apache Hadoop [26] que permite la utilización del paradigma de MapReduce en un cluster distribuido.

	Incremento 2011-2016
Intercambio de archivos	26%
Vídeo por Internet	34%
Internet, email y datos	35%
Juegos online	52%
Voz sobre datos (VoIP)	3%
Total	32%

Tabla 2 - Incremento porcentual del tráfico de usuarios.
Fuente Cisco VNI, 2014 [10]

Por un lado tenemos menos tratamiento del vídeo en plataformas de Big Data en comparación con los tratamientos de los textos procedentes de páginas web o blogs. Por otro lado existen algunas herramientas que permiten el tratamiento del audio para su reconocimiento. En este proyecto se desea evaluar la posibilidad de tratar grandes volúmenes de vídeo y/o audio, procedentes de las capturas de las transmisiones online y verificar que un modelo Big Data se adecua a esta tarea. Se pretende desarrollar una aplicación que sea capaz de capturar parcialmente vídeos de algunas cadenas de televisión que emiten por *streaming* a través de Internet y generar algún conocimiento que permita su uso por un usuario que no tenga acceso al vídeo original y/o desee información de los vídeos tratados sin tener que visionarlos en su totalidad. Un ejemplo de este tipo de información puede ser una nube de palabras que permita visualizar un histograma de frecuencia de apariciones de ciertas palabras en dicha información de audio.

Durante el apartado 2 de esta memoria se explican las valoraciones previas que se realizaron antes de comenzar el desarrollo del proyecto y el esquema inicial del tratamiento. En el apartado 3 se comentan las herramientas, formatos y estructuras de datos que serán necesarias para la realización del proyecto. En el apartado 4 se muestran los detalles de la implementación incluyendo la división de los procesos en tareas MapReduce y en el apartado 5 se explican los tiempos empleados en las capturas realizadas de las ejecuciones descritas en la sección anterior y algunas valoraciones temporales/espaciales de su posible ejecución. Se finaliza con una sección de conclusiones y trabajos futuros así como algunos anexos donde se da información detallada de los códigos desarrollados para la realización del trabajo.

2 Tareas Previas

2.1 Definición del modelo de tratamiento

Después de un estudio previo de las tareas que se desean realizar en este proyecto se llegó a la conclusión que el modelo del tratamiento, que se quiere verificar, se divide en las siguientes etapas:

Etapa 1. Captura.
El proceso de captura permitirá el almacenamiento de los ficheros multimedia en los almacenes de información para su posterior tratamiento. Se pretende capturar los contenidos multimedia a partir de los flujos de *streaming* emitidos por las cadenas de televisión en un sistema de almacenamiento permanente que permita un tratamiento asíncrono de los contenidos.

Etapa 2. Adecuación y normalización de los ficheros.
Es evidente que los ficheros deben de tener un formato adecuado para su posterior tratamiento. Para ello, se convertirán en un formato que posibilite que los ficheros sean tratados de forma homogénea en las etapas posteriores. Como complemento a la adecuación del formato, se pueden realizar otras tareas que permitan un mejor reconocimiento, la adecuación del volumen o la eliminación de ruido pueden llegar a incrementar el éxito de las fases posteriores.

Etapa 3. Diarización y generación de procesos.
Para el tratamiento correcto de los ficheros estos deben tener tamaños inferiores a los de captura. Con la diarización se dividen los ficheros por hablante y se reconocen los espacios sin habla. De esta forma, se generan las porciones que luego serán reconocidas vocalmente. Este proceso de generación de porciones de datos encaja perfectamente con el propósito de tratamiento de los datos en un sistema de procesamiento distribuido de tipo HTC (*High Throughput Computing*).

Etapa 4. Tratamiento de ficheros multimedia: reconocimiento.
En este proceso se realizará el proceso de reconocimiento vocal del sonido y su almacenamiento en ficheros de texto. Para ello, es necesario utilizar herramientas de reconocimiento ya desarrolladas por otros autores y que hacen inevitable la utilización de un modelo de reconocimiento entrenado para las tareas asignadas.

Etapa 5. Generación de conocimiento
Utilizando como base los datos proporcionados por la diarización y el reconocimiento vocal, estos resultados de fases anteriores se tratarán estadísticamente para su mejor comprensión por parte de los usuarios finales. El tratamiento histórico de estos datos supone una parte esencial ya que nos permite ampliar las dimensiones disponibles de los datos con una nueva dimensión temporal.

Etapa 6. Presentación de los resultados
Se buscarán medios de representación de los datos obtenidos en la fase anterior para que sean fácilmente comprensibles por el usuario final. Se sugiere, como una representación inicial, la nube de palabras para permitir una visualización rápida y eficaz de diferentes contextos.

En la "Figura 2 - Diagrama inicial de etapas del proceso" se pueden observar las etapas anteriormente descritas donde se distinguen dos tipos de líneas. Las líneas naranjas, hasta la etapa 4, corresponden a los tratamientos de formatos binarios (audio o audio/vídeo) mientras que las líneas grises, de la etapa 4 en adelante, denotan una trasferencia de datos tipo texto. La etapa 4 (reconocimiento) es la que marca la frontera entre los dos tipos de datos tratados.

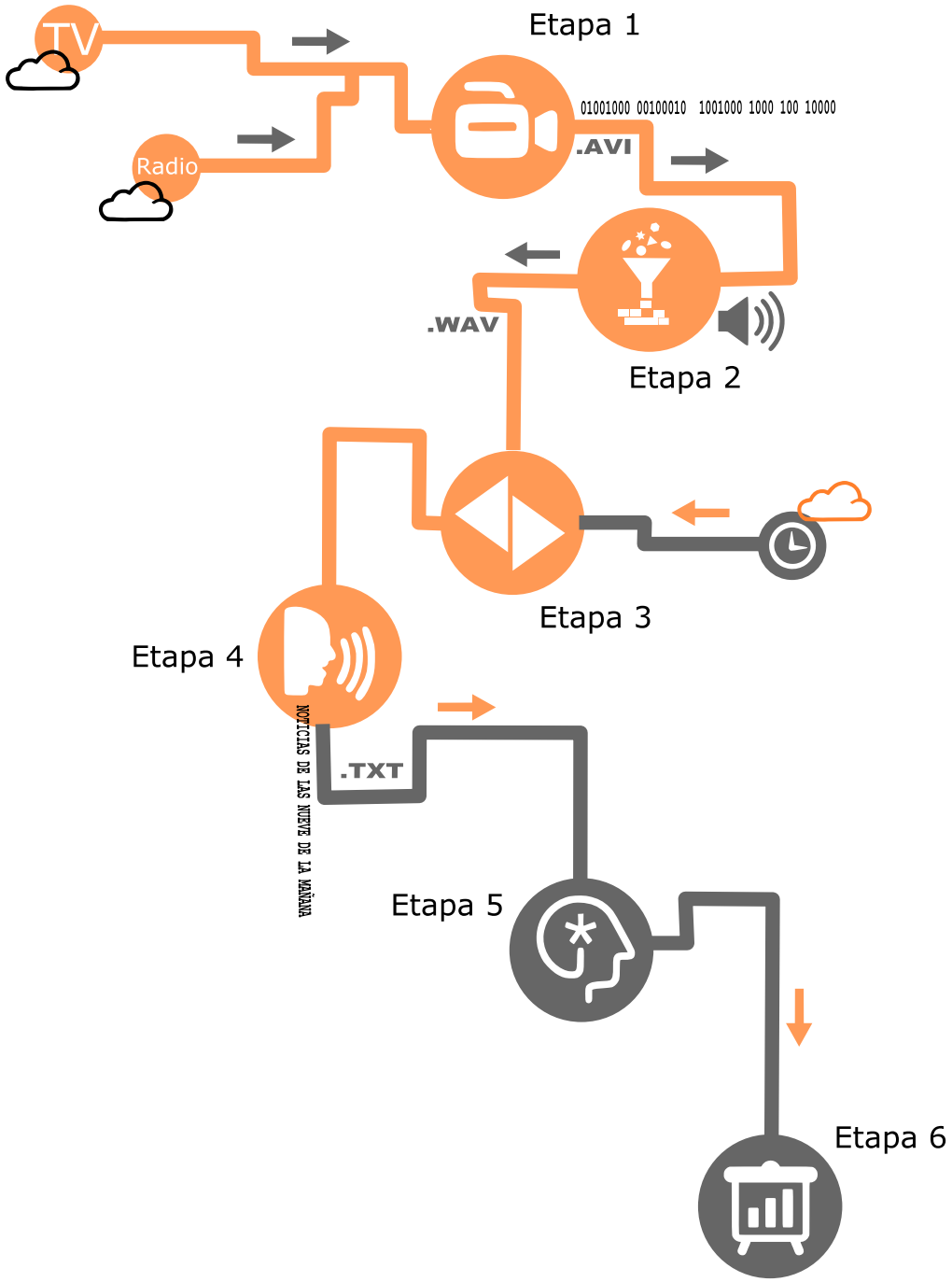

Figura 2 - Diagrama inicial de etapas del proceso

2.2 Valoración de los tamaños iniciales de los ficheros de captura

Previamente al diseño de los procesos y como parte del análisis desarrollado, se ha valorado el tamaño de los datos que deben tratarse y se muestran en la "**Tabla 3 - Estimación en GB del tamaño de captura de 10 emisoras**

	TV	Radio	Total
15 Min.	0,9	0,7	1,6
1 Hora	3,4	2,9	6,4
1 Día	82,5	70,3	152,8
1 Semana	577,5	492,2	1.069,7
1 Mes	2.475,0	2.109,4	4.584,4
1 Año	30.112,5	25.664,1	55.776,6

Tabla 3 - Estimación en GB del tamaño de captura de 10 emisoras

La estimación se ha confeccionado en base a una captura de 10 emisoras de TV y 10 de audio con una duración de 15 minutos en los dos casos. La captura de TV ha sido en formato comprimido Flash Video que utiliza Sorenson Spark (FourCCFLV1) una variante propietaria del codec H.263. Este formato es bastante utilizado para las emisiones de video en Internet. Las capturas de radio se estimaron en formato sin compresión .wav que es un formato que necesitará poca conversión para su reconocimiento. Es por ello que el tamaño de la captura de video y el de audio son tan similares.

2.3 Entornos de desarrollo. Hadoop Distribution

Hadoop es la herramienta seleccionada para la realización de los desarrollos de este proyecto debido a su amplia implantación en el mundo del Big Data y su capacidad de tratamiento de grandes volúmenes de información. En un cluster Hadoop podemos diferenciar dos grandes funciones: la ejecución de trabajos y la gestión de los datos. Hadoop se compone, en esencia, de dos tipos de nodos. Un Master que tiene la responsabilidad de administrar los trabajos y los metadatos del sistema de almacenamiento y un segundo tipo que tiene la responsabilidad de almacenar bloques de datos y ejecutar porciones de los trabajos. En la sección "3.6 - Hadoop" se explica más detalladamente el funcionamiento del sistema de gestión de ficheros y la ejecución de trabajos.

Antes de la implementación hay que tener en cuenta el entorno que se va a utilizar. Una primera posibilidad es la realización de la instalación en un sistema sin ningún software previo y la segunda es la utilización de una compilación de programas empaquetados en una distribución totalmente operativa o fácilmente instalable. Entre las diferentes posibilidades, en esta segunda opción, nos podemos encontrar con las siguientes:

- **Cloudera**. [11] Cloudera Inc. es una empresa norteamericana de software cuyo producto más destacado es la distribución de Hadoop CDH (Cloudera Hadoop Distribution). Fue fundada en 2008 por los desarrolladores de Big Data en Facebook, Google, Oracle y Yahoo.
- **Hortonworks**. [30] La empresa fue fundada en 2011. Es la única distribución que no incluye software propietario adicional. En la actualidad se distribuye de forma libre en su versión 2.0. Se trata de una *spin off* de Yahoo.

- **MapR**. [43] Esta empresa californiana proporciona tres distribuciones: M3 (versión libre), M5 (versión de pago) y M7 similar a la versión M5 pero mejorada. Esta es la versión que se utiliza en Amazon Elastic MapReduce (EMR).
- **IBM Infosphere BigInsights**. [31] Es la propuesta de distribución de IBM para el mundo de Big Data. Reúne, lo mismo que las demás distribuciones, numerosas herramientas para "*descubrir y analizar ideas de negocio ocultos en grandes volúmenes*" [51]. En los siguientes párrafos se detallan algunas características que han sido necesarias utilizar en el desarrollo del proyecto.

Otras posibles opciones son la utilización de entornos en Cloud como pueden ser BlueMix [6] de IBM, Amazon Web Services (ya hemos comentado que utilizan la distribución de MapR), o Azure de Microsoft.

La distribución seleccionada para realizar las tareas de desarrollo ha sido IBM InfoSphere BigInsights v3.0. Esta distribución nace con la idea de mejorar la experiencia empresarial y hacer más simple la utilización del entorno de Big Data Hadoop. Pretende y logra, en algún caso, mejorar las tecnologías *open source* añadiendo una mejor administración, desarrollo, seguridad y soporte. Como resultado se obtiene un entorno más amigable para proyectos a gran escala y complejos. En el proyecto que nos ocupa se han mejorado algunos aspectos que paso a destacar y que resultan importantes para el desarrollo que se hace en este proyecto.

- Entorno de desarrollo. Proporciona un Entorno de Desarrollo Integrado de fácil instalación y mantenimiento, basado en Eclipse [15] Juno. Actualmente existen dos versiones posteriores de Eclipse: Keppler y Luna. Incluye posibilidades de ejecución en local con un único nodo y mediante la utilización de su API la emulación de un sistema de ficheros HDFS en local.
- Inclusión y tratamiento de nodos. Se ha creado una interfaz para ampliar el cluster Hadoop con nuevos nodos y su adecuación a las tareas que se les desean asignar. El sistema de inclusión es bastante automatizado.
- Interface gráfico para HDFS [27]. Se ha incluido una interfaz gráfica para la edición, visualización y operación del sistema de ficheros de Hadoop. No permite la creación de ficheros con el editor, aunque sí de directorios. Facilita la subida de ficheros locales al sistema de archivos HDFS en el cluster Hadoop y la bajada en sentido inverso.
- Publicación de código. Se han incluido métodos sencillos para poder subir al cluster Hadoop los desarrollos realizados (publicar) y posteriormente se puede realizar el despliegue de estos desarrollos previa a su ejecución.
- Gestión de los workflows. Se ha incluido un gestor de workflow mediante la utilización de Oozie. Esta herramienta permite la realización de tareas complejas mediante la unión de varios procesos MapReduce, shell scripts, Pig shell, etc. Oozie por su parte ya tiene un gestor gráfico, aunque en la distribución de BigInsights está desactivado. Se ofrecen similares funcionalidades en el portal de BigInsights (ver "4.1 - Workflow Administrator: Apache Oozie") .
- Inclusión de herramientas para el tratamiento estadístico. R [33] y BigR [5]. Aunque no vienen desplegados en un principio no es complicada la tarea de realizar una instalación de ninguna de las dos. R constituye una de las herramientas estadísticas más utilizadas en la actualidad (ver 3.5.1- Software para el tratamiento estadístico: R"). BigR es una librería propietaria de IBM que permite la ejecución de ciertos tratamientos de R en un cluster Hadoop.

3 Herramientas y estructuras de datos utilizadas

3.1 Captura y normalización de las emisiones

Con el aumento del ancho de banda de las conexiones a Internet, se ha permitido también que los contenidos multimedia, reservados anteriormente a la difusión por vía aérea o a las líneas de TV vía cable dedicado, se puedan acceder por este medio. Cada vez tenemos más emisoras de radio y TV que emiten sus programaciones vía Internet sin ningún tipo de restricción para su consumo desde dispositivos móviles o fijos. Son estas emisiones las que centrarán las capturas que se realizarán en este proyecto aunque no se descarta que en un futuro las fuentes de datos puedan ser más variadas (dispositivos de captura, plataformas web de vídeo, etc.). Emisoras de televisión como a3, La Sexta, TV3, 24h, RiberaTV o radios generalista como SER Valencia, COPE, RNE1, RNE5 o Radio Exterior realizan emisiones por Internet que se pueden capturar. El procedimiento de captura de las informaciones se basa en la ejecución de un script bash.

Con el fin de acotar, el problema el grupo de emisoras tratadas se restringe a aquellas de habla castellana. El área geográfica se centra en las emisoras españolas y, en caso de existencia de emisiones regionales, a las disponibles para la Comunidad Valenciana.

Es común que los formatos de almacenamiento de los datos transmitidos utilicen algún sistema de compresión de los datos. De esta forma las emisiones de radio suelen realizar la emisión en formato MPEG-1 o 2 Layer III (más conocido como MP3) mientras que las televisiones utilizan formatos propietarios de Adobe Flash Player.

Ya en [37] se realiza una propuesta para la transformación de contenido multimedia mediante la utilización de un servicio MapReduce, por lo tanto no se centrarán los trabajos en esta área. En el "Anexo I - Ejemplo de implementación para el bloque 1" está disponible un script, que mediante los comandos *mplayer* [45] y *ffmpeg* [18], permite la captura de los contenidos y su transformación en un formato de tratamiento adecuado. La etapa 1, del proceso de tratamiento, realiza la captura de las emisiones en ficheros propios de la emisión. De esta forma los ficheros capturados de TV tienen formato diferente a los de radio. La etapa 2 realiza la conversión del formato de fichero capturado en la etapa anterior para convertirlo en un fichero con formato wav con las características que muestra la captura de la Figura 3.

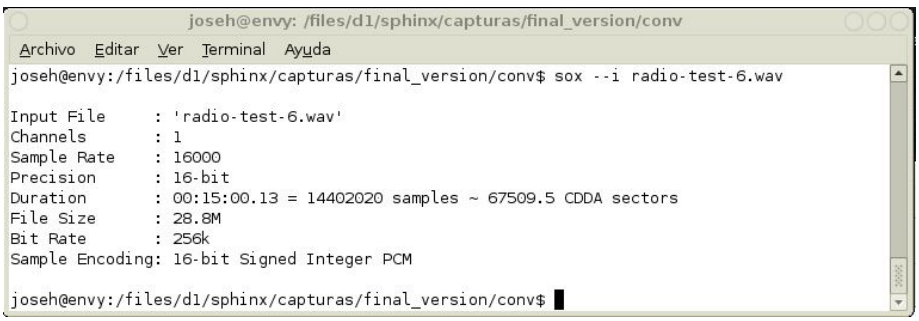

Figura 3 - Formato de conversión para los ficheros de audio

Como parte adicional a este script se ha incluido la generación del *sequence file* mediante la utilización del código TarToSeqFile [53]. Este código realiza la generación de ficheros sequence file desde ficheros .tar lo que evita tener que desarrollar nuevo código con la misma

funcionalidad. El *sequence file* es uno de los medios que tiene Hadoop para almacenar grandes volúmenes de información en formato binario y que se detalla en la sección "3.2.1.1 - *Sequence files*".

Ya que el entorno de desarrollo que se va a utilizar es un entorno de tratamiento batch es necesario realizar una propuesta de tiempos de captura para las emisiones. Esta se definirá como el periodo de tiempo que se captura en un único fichero físico y que una vez finalizada comenzará su tratamiento. El periodo seleccionado inicialmente es de 15 minutos. Por lo tanto, dispondremos de 4 ficheros por hora para su tratamiento.

3.2 Almacenamiento de la información

Tras las capturas y la normalización de las emisiones, que corresponde con la finalización de la fase 2, se debe disponer de un almacenamiento adecuado, con el fin de lograr que el gran volumen de información que se debe procesar quede disponible para los procesos posteriores que continuarán el tratamiento.

Por un lado, debemos tener en cuenta que el almacenamiento en ficheros individuales, correspondientes a capturas de pequeño tamaño, no es una buena solución en Hadoop. Se entiende que un fichero es pequeño si su tamaño es considerablemente más reducido que el tamaño del bloque de Hadoop, en nuestro caso 128 MB. Pero también hay que tener en cuenta que estos ficheros no pueden representar, de forma nativa, la información temporal y en nuestro caso es una información primordial ya que muchos de los tratamientos que se realizarán con posterioridad tienen la dimensión del tiempo como uno de sus ejes principales. Por ello, se realiza una propuesta de estructura de almacenamiento que soporta el volumen de información previsto al mismo tiempo que incluye su información temporal

En el caso de los ficheros de tamaño reducido, las dos soluciones que propone Hadoop para su almacenamiento dentro de HDFS son:

- *Hadoop Archives* (HAR files). Supone el tratamiento de todos los ficheros que existen en un directorio como si fuera un único fichero. Esto es apropiado para un único fichero que va creciendo progresivamente y al que se le van añadiendo porciones. El fichero en definitiva no es pequeño sino que se compone de pequeñas partes. Un ejemplo de esto son los ficheros de log.
- *Contenedores de alto nivel*. Son adecuados para almacenar ficheros inherentemente pequeños. Un ejemplo sería un almacén de fotografías con tamaño reducido.

Los formatos de almacenamiento de la información propios de Hadoop se describen en las siguientes secciones.

3.2.1 Contenedores de alto nivel en Hadoop

Hadoop ha desarrollado un grupo de contenedores de alto nivel con el fin de almacenar y procesar información de tipo binaria. Dentro de esta categoría podemos distinguir dos tipos:

- Los *sequence files* son un tipo de ficheros de datos propios de Hadoop que almacena información en formato key/value y están codificados de forma binaria.

- Los *Map File* son *sequence files* ordenados con un fichero adicional con índice que permite las búsquedas por la clave de las parejas key/values.

Con vista a realizar una selección del tipo de fichero que se va a utilizar para el almacenamiento de los ficheros de audio que se procesarán, se han estudiado los contenedores de alto nivel observandose que son de similares características pero en el caso de los *Map File* se añade información adicional para la localización rápida de parejas key/value.

3.2.1.1 *Sequence files*

En este tipo de ficheros se encuentran tanto los propios datos como los metadatos, permitiendo su transporte y localización. Una de las características principales de este tipo de ficheros es que son "*splittables*" (que se pueden fragmentar). El nivel de compresión puede ser de tres tipos:

- Sin compresión.
- *Record Compressed*. La compresión se produce a nivel de cada par key/value.
- *Block Compressed.* La compresión se produce a nivel de bloque (grupos de key/values).

Independientemente del nivel de compresión que se utilice siempre está disponible una cabecera (*header*) que contiene información para la lectura de fichero.

Los principales inconvenientes son su accesibilidad única y exclusivamente mediante la API que proporciona Hadoop y que la definición inicial del key/value define los tipos de lecturas que deben realizarse en el resto de fichero.

El contenido del fichero se indica en la cabecera. Una vez fijada esta estructura por la cabecera no se puede modificar en el resto del fichero, incluido el tipo de compresión y el codec de compresión que se va a utilizar. De una forma visual la cabecera de todos los *sequence file* contienen la siguiente información:

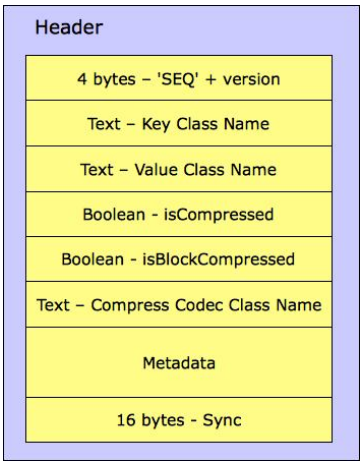

Figura 4 - Composición de la cabecera en un *sequence file*

La estructura es igual para los ficheros comprimidos a nivel de registro o los que no tienen compresión, diferenciándose en el contenido de los bloques que constituyen cada uno de los registros.

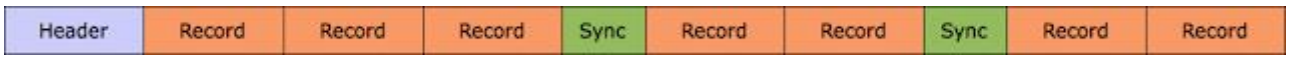

Figura 5 - Estructura de los *sequence file* (sin compresión o compresión a nivel de registro).

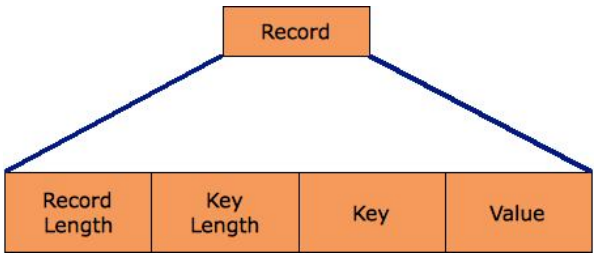

Figura 6 - Estructura del registro en los *sequence file* sin compresión

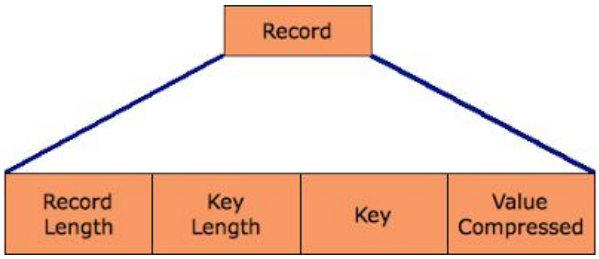

Figura 7 - Estructura de los registros *sequence file* con compresión por registro

Con la compresión a nivel de Bloque (Block Compressed) la estructura del fichero se modifica y pasa a ser la siguiente:

Figura 8 - Estructura de los *sequence file*s (con compresión a nivel de bloque).

Siendo los bloques de la siguiente forma:

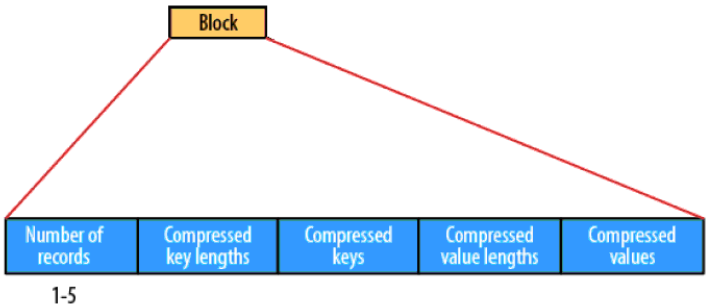

Figura 9 - Estructura del registro en los s*equence file* con compresión por bloque

3.2.1.2 Map Files

Los *Map Files* (ver "Figura 10 - Estructura de un *Map File*") son un directorio que contiene dos archivos *sequence file*: El fichero de datos (“/data”) y el índice (“/index”). El fichero de datos contiene toda la información de la clave con la información del valor a continuación. Esta condición se verifica en el momento de la creación.

El fichero de índice se compone con la clave y un dato de tipo *LongWritable* que contiene la posición de inicio del primer byte del record del *sequence file*. No contiene todas las claves, pudiéndose indicar el intervalo (*indexInterval*). El índice se mantiene en memoria, por lo tanto si el fichero es muy grande, el *indexInterval* debe ser acorde para que el volumen total del fichero /index se pueda almacenar en su totalidad.

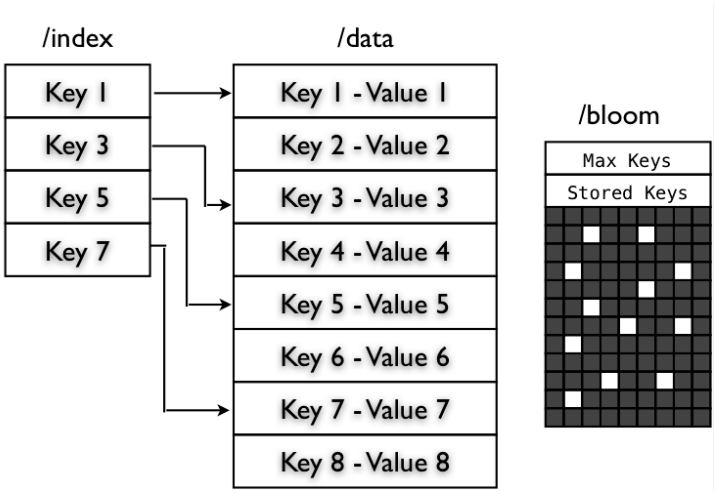

Figura 10 - Estructura de un *Map File*

Existen variaciones de esta estructura como pueden ser los *SetFile*, *ArrayFile* o *BloomFile*. Este último incorpora un tercer fichero (/bloom).

La utilización de algún tipo de índice (/index o /bloom) facilita la localización rápida de parejas key/value dentro de ficheros extensos. Ya que el tratamiento de los ficheros que se realiza es de tipo secuencial por bloques no es necesario el mantenimiento de índices de busqueda, por tanto se utilizará el *sequence file* sin información de índices.

3.2.2 Estructura de almacenamiento de datos e información

Para poder mantener el gran volumen de datos que se pretende capturar y/o generar y al mismo tiempo favorecer el cometido de Hadoop se han procedido a diseñar un almacenamiento de datos en ficheros dentro de sistemas de archivos HDFS [27].

Se ha implementado, bajo el directorio "newsASR", una estructura multinivel de directorios que permite el almacenamiento de los datos que se generan por los procesos implicados en el tratamiento de una forma ordenada, lógica y fácilmente accesible. Los niveles son los siguientes:

Directorio "run". Información sobre los ficheros XML que necesita Oozie para la puesta en marcha de los diferentes procesos.
Directorio "temp". Almacenamiento de ficheros temporales necesarios para los tratamientos e información final de los procesos de generación de conocimiento.
Directorio "lib". Ficheros que se hacen disponibles a los procesos MapReduce para poder realizar las tareas asignadas.
Directorio "data. Contiene la información capturada y generada por los procesos**.**

- Nivel 1. Año
- Nivel 2. Mes
- Nivel 3. Día
- Nivel 4. Hora
- Nivel 5. Minutos. Este nivel podrá tener el nombre 00, 15, 30 o 45 asociado cada uno a los datos que se comenzaron a capturar en ese minuto. Además poseerá el fichero *sequence_file.sf* que corresponde al fichero binario de Hadoop con la información de las

capturas normalizadas que se han realizado en el periodo año/mes/día y hora/minutos donde este situado.

- Nivel 6. Directorio con el nombre output que contendrá información generada por los procesos MapReduce.
- Nivel 7. Puede tener los siguientes directorios
    - en stage1. Contendrá la información intermedia correspondiente a la etapa 4 (se verá más adelante)
    - en stage2. Contendrá la información generada correspondiente a la etapa 5 (se explicará más adelante)

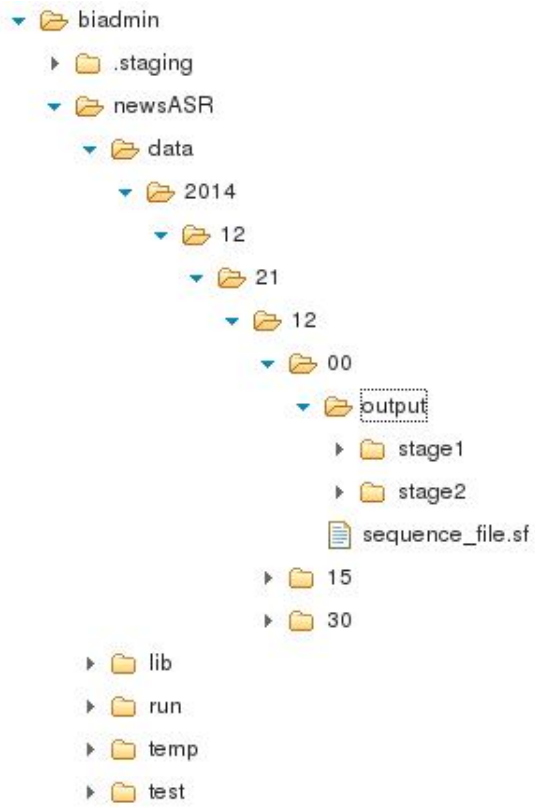

Figura 11 - Propuesta de estructura de directorios para el almacenamiento de datos

3.3 Diarización

El tratamiento de la información por medio de Hadoop, lo mismo que otros procesos de ejecución paralela, implica la necesidad de dividir la información de modo que esta se pueda tratar de manera independiente en cada uno de los procesos/máquinas o nodos de ejecución (ver "Figura 12 - Paralelismo de datos de tres Procesos en Ejecución"). Para ello se utiliza una aproximación de paralelización de tipo SPMD (*Single Program Multiple Data*) donde se ejecuta el mismo código sobre un conjunto de datos diferente.

Por el mero hecho de realizar la captura de una emisora ya hemos realizado una diferenciación de los datos con respecto a la captura de otras emisiones, pero el grado de paralelización que se obtiene por esta vía es muy reducido siendo como máximo el número de fuentes multimedia capturadas.

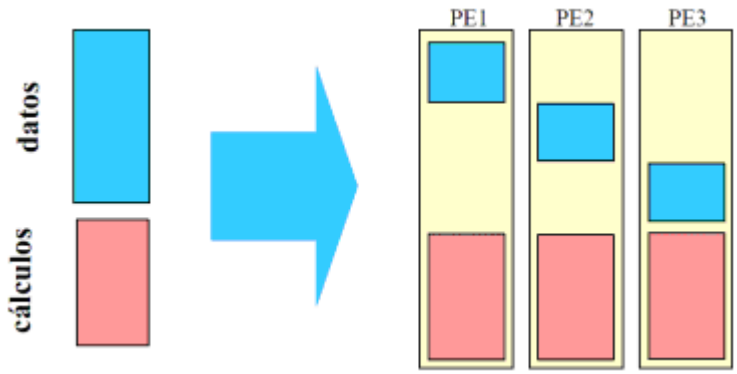

Figura 12 - Paralelismo de datos de tres Procesos en Ejecución

Por ello, se han buscado vías alternativas de aumentar el grado de paralelismo de los procesos que deben tratar los datos. Las alternativas han sido:

- Por **división temporal**. Realizar *splitting* de los datos en función únicamente del tiempo. Se puede reducir el tiempo máximo de captura haciendo porciones más pequeñas basadas en un espacio de tiempo fijo o variable. El problema se plantea en el desconocimiento del lugar donde debe realizarse el corte ya que lo ideal sería que fuera en algún silencio, deconociendose el lugar con anterioridad.
- Por **división espacial**. Realizar un corte de los ficheros dependiendo del espacio ocupado. Por ejemplo cortamos cada 500MB. Tiene el mismo problema que el caso anterior, realizando corte en lugares inadecuados.
- **Diarización** o localizar los márgenes del habla en los ficheros de audio. La diarización (*diarization* o *speaker diarisation*) es el proceso de particionado de un *stream* de audio en segmentos homogéneos dependientes de la identidad del hablante, en otras palabras "*who spoke when*?" [38]. El principal problema puede surgir por la no existencia de una duración homogénea. Esto es, los tramos que pueden aparecer de un hablante u otro o los espacios en blanco pueden ser muy grandes o muy pequeños lo que implica un particionamiento que no se puede controlar en un inicio. Esta es la opción seleccionada para aplicarla en el proyecto.

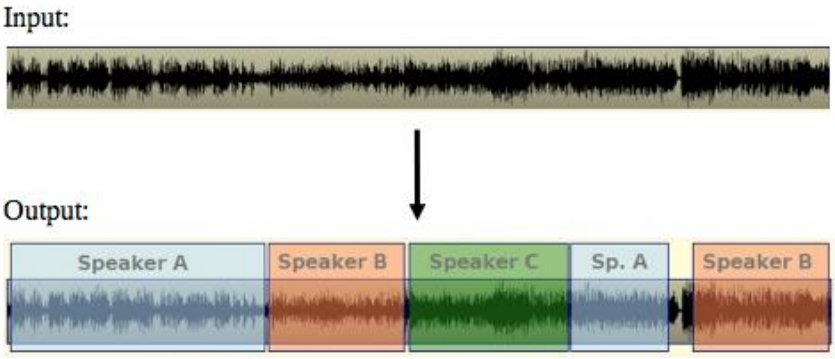

Figura 13 - Proceso de diarización

Las posibles opciones que surgen para la selección de un software de diarización son:

- LIUM_Spkdiarization [40] es un software escrito en Java dedicado a la diarización y clusterización, diseñado en el Laboratoire d'Informatique de l'Université du Maine [39]. Esta herramienta incluye un conjunto de herramientas para la identificación del hablante, detección de género, diarización simple o cruzada, etc. El toolkit está entrenado con el corpus ESTER2 razón por la cual se obtienen los mejores resultados sobre el broadcast de noticias francesas. Esta especialmente optimizado para radio y TV.

- SHoUT [52], que corresponde con el acrónimo en Alemán de Investigación en Reconocimiento Vocal en la Universidad de Twete. Al igual que otras opciones está desarrollado en código abierto y está escrito en C++ para plataformas Linux.

- ALIZE. LIA/RAL. [2][7] Es un desarrollo de la Universidad de Avignon con licencia GNU/LGPL. Se implementó en C++.

- AudioSeg [1][22], es un toolkit dedicado a la segmentación y clasificación de streams de audio. Esta desarrollado en C para sistemas operativos Linux bajo licencia GPL. Todo ello ha sido desarrollado en el INRIA, Institut National de Recherche en Informatique et Automatique, que es el ente público francés de investigación especializado en la computación y las matemáticas aplicadas. Implementa tanto la detección de silencios, la clusterización como la segmentación.

La elección de la herramienta no ha sido una cuestión sencilla. Se ha seleccionado en un principio la que se adaptaba más al entorno de desarrollo, pero luego se ha abierto a la posibilidad de utilizar cualquiera que se considere adecuada. Se optó por LIUM_Spkdiarization por las valoraciones generales que se han podido hallar tanto en foros como en blogs especializados en los que se explicaba su facilidad de uso y el desempeño que proporcionaba la herramienta.

Aunque otro de los puntos que influyó en la elección de la herramienta de diarización es su desarrollo en el lenguaje Java, con vistas a poder integrar facilmente la diarización en el proceso Map, se ha programado una solución final a modo de wrapper que permite la ejecución de un script bash y la devolución de resultados por medio de un fichero que es tratado en el mismo proceso Map. De esta forma sería incluso viable la realización de varias diariazaciones con diferentes herramientas y seleccionar la más adecuada.

3.4 Reconocimiento

ASR (Automatic Speech Recognition), SR (Speech Recognition) o STT (Speech To Text) son un conjunto de sistemas que permiten convertir un stream de audio a su transcripción en texto.

Como complemento a los sistemas ASR, tenemos los sistemas de reconocimiento vocal o identificador del hablante. En vez de realizar una transcripcion del audio, estos sistemas pueden facilitar la realización de un sistema de seguridad (autenticación o verificación de hablante) o bien mejorar y facilitar los procesos de ASR ya que están mejor definidos los sonidos que se deben tratar.

Entre las diferentes posibilidades que se han evaluado para el reconocimiento están las siguientes:

- Julius. [34] La principal plataforma de ejecución es Linux y otros sistemas operativos Unix. Las versiones más recientes funcionan en Linux y Windows (cygwin/mingw) y también como interfaz de programación de aplicaciones de voz SAPI (Microsoft Speech API). Se distribuye con el código fuente y licencia abierta. Ha evolucionado desde su primera versión en 1997 pasando por varios proyectos. La última versión es la rev.4.3.1 de enero de 2014. Como características destacables podemos mencionar su bajo consumo de memoria y su adecuación a sistemas de tiempo real.

- CMU Sphinx [56] es un sistema desarrollado en Java para el reconocimiento vocal. Proporciona interfaces de alto nivel y está diseñado de forma modular con el objetivo de facilitar su flexibilidad. Ha sido implementado por la Universidad Carnegie Mellon, Sun Microsystems Lab y Mitsubishi Electric Research Laboratories (MERL). Actualmente se encuentra en la versión 4.

- TLK (transLectures-UPV Toolkit package) Es un conjunto de herramientas para la realización de Automatic Speech Recognition (ASR) desarrollado en la Universitat Politècnica de València - UPV con C/C++. Dispone de su propia herramienta de diarización.

Para la realización de una aplicación operativa y la evaluación de tiempos se ha seleccionado Sphinx4 que es fácilmente empleable en la aplicación principal además de ser una de las opciones más ampliamente utilizadas. Un reconocedor desarrollado en Sphinx necesita para su correcto funcionamiento de un modelo entrenado compuesto por:

- **Modelo de lenguaje**. Se proporciona una estructura del lenguaje a nivel de palabras. Con estos modelos se limita el espacio de búsqueda en las tareas de reconocimiento lo que hace que el lenguaje soportado sea mayor y el tiempo utilizado sea más reducido. Las implementaciones generalmente son de dos tipos [8]:
    - Modelos basados en gramáticas. Este modelo es utilizado cuando el número de palabras es muy reducido.
    - Modelos de lenguaje Stochastic (N-GRAM) que es un modelo probabilístico en base a las palabras que se dispongan hasta el momento.
- **Diccionario**. Proporciona la pronunciación para las palabras encontradas en el modelo del lenguaje.

- **Modelo Acústico.** Es la parte del reconocedor que contiene la variabilidad acústica de la señal de entrada. Los modelos acústicos fonéticos se basan en modelos de Markov (HMM - Hidden Markov Model).

Debido a la necesidad de este modelo entrenado no es posible realizar un reconocimiento válido sobre las emisiones capturadas que tenemos como objetivo. En nuestro caso, esto no representa un impedimento para continuar el estudio ya que se centra en las posibilidades que surgen para su tratamiento en los entornos de BigData, dejando el reconocimiento vocal para otras áreas de la computación que quedan fuera del ámbito del proyecto. Si que se puede utilizar modelos con limitaciones. Uno de ellos es el generado a partir del corpus público de Voxforge [55]. Este conjunto de ficheros de audio libre nos servirá para realizar juegos de prueba construyendo ficheros de similares características, duración y formato a los que se lograrían en el proceso de captura de emisiones por Internet.

Sphinx Train es el conjunto de herramientas (programas, scripts y documentación) desarrolladas para las diferentes versiones de Sphinx con el fin de permitir el aprendizaje del sistema de diferentes idiomas por parte de cualquier persona interesada en el desarrollo de herramientas basadas en el reconocimiento vocal.

Con el fin de independizar el proceso de reconocimiento de la herramienta utilizada se ha modelado el proceso como un script bash. El proceso requiere un fichero wav con el audio que se debe reconocer. Posteriormente se produce la invocación del script (./run.sh) y como resultado se obtiene un fichero que contiene las palabras reconocidas. Con independencia de estas interfaces el proceso Java de Hadoop no tiene ninguna relación con el proceso de reconocimiento que se está ejecutando. De esta forma su adaptabilidad es mayor y se puede sustituir por otro proceso de reconocimiento que se considere más adecuado.

El mismo sistema indicado en el párrafo anterior permite el tratamiento multi-idioma. Esto posibilitaría la utilización de un modelo diferente, entrenado para cada uno de los streams, permitiendo trabajar con variedad de idiomas. La utilización de diferentes modelos vocales entrenados para diferentes situaciones puede suponer una mejora significativa en el reconocimiento.

De la misma forma que en la sección de diarización se ha propuesto realizar varias diarizaciones con el fin de seleccionar aquella que sea mejor, también sería posible realizar varios reconocimientos con el fin de analizar aquellos que han resultado mejores, tanto con modelos vocales diferentes para un mismo idioma como para otro idioma con distintas herramientas de reconocimiento.

Corpus o *Speech Corpus* [55] es una base de datos de ficheros de audio y sus transcripciones. Este almacén es utilizado para entrenar un modelo acústico que puede ser utilizado por los sistemas de reconocimiento vocal. Se pueden dividir en dos categorías: *Read Speech* (lecturas de libros, noticias de radio y TV, listas de palabras) y *Spontaneous Speech* (Diálogos, Narrativas, etc.). Pero también podemos encontrar Corpus de hablantes no nativos que contienen información de hablas con acento extranjero. De igual manera se utilizará el *Corpus* como fuente para generar ficheros de prueba que sean comprensibles por el sistema de reconocimiento. De esta forma juntando varios ficheros del *corpus* se obtiene un *sequence file* similar al que se obtendría en el proceso de captura pero con el añadido que sea reconocible por nuestro sistema.

Por último vamos a definir algunas medidas que son necesarias para la evaluación del sistema de reconocimiento.

WER (*Word Error Rate*). Supongamos que se posee un texto procedente del reconocimiento de un audio con longitud N palabras. De ellas I palabras han sido insertadas (no existen originalmente), D palabras han sido eliminadas (no aparecen el texto) y S palabras han sido sustituidas (se ha cambiado una palabra por otra). La tasa de error queda definida de la siguiente forma:

$$WER = \frac{I + D + S}{N}$$

Fórmula 1

Precisión (*Accuracy*). Es similar al WER pero sin tener en cuenta las inserciones.

$$Accuracy = \frac{N - D - S}{N}$$

Fórmula 2

Speed. Se define en función del tiempo del audio original (T_{reales}) y el tiempo que tarda en ejecutarse el trabajo de reconocimiento (T_{job}). Si suponemos que el audio tiene 2 horas y el proceso de reconocimiento tiene una duración de 6 horas tendremos un sistema 3xRT. La utilización de un sistema superior a 1xRT implica que la duración del reconocimiento es mayor en tiempo que el tiempo del audio. Si quisiéramos un sistema en tiempo real (*Real Time*) deberíamos tener un sistema con valor igual o inferior a 1.

$$Speed = \frac{T_{job}}{T_{reales}} xRT$$

Fórmula 3

Tanto el WER como la precisión suelen presentarse a nivel de porcentaje.

3.5 Extracción del conocimiento

Una vez finalizados los procesos de reconocimiento del audio por medio de la utilización de técnicas de Big Data utilizando el soporte que para ello proporciona Hadoop, se plantea la presentación de los datos obtenidos de una forma más próxima y entendible al usuario final. En un principio el proyecto se vio como la realización de un "*informe de prensa no escrita*" y la utilización de un sistema de información gráfico que permitiera la visualización de los datos de una forma más asimilable por el usuario final. La selección de un sistema de representación adecuado puede llegar a proporcionar el éxito al sistema que de otra forma puede no ser utilizado.

Las nubes de palabras o nubes de etiquetas son un método de representación sencillo y ágil de la información donde los "tags" mostrados tienen diferentes colores o tamaños en función de la importancia. Una primera aproximación para exponer los datos extraídos del proceso es la generación de una nube de palabras que plasme de forma visual el número de apariciones de una palabra en un periodo de tiempo reducido y definido.

Independientemente de la necesidad de realizar la nube de palabras como método básico para la representación de la información obtenida, se sugiere la oportunidad de realizar otros tratamientos más profundos de los datos procesados, centrándose especialmente en la variable tiempo. De esta manera se ha llegado a la utilización de un software con suficientes características para el tratamiento y representación que se describe en el punto 3.5.1.

3.5.1 Software para el tratamiento estadístico: R

R [33] es un lenguaje de programación con una implementación *open source* para tratamiento estadístico y generación de gráficos. Es ampliamente utilizado en procesos estadísticos y de datamining. Fue creado por Ross Ihaka y Robert Gentleman en la Universidad de Auckland, Nueva Zelanda. Está liberado con licencia GNU -General Public Licence- lo que ha hecho que sea una de las herramientas de tratamiento estadístico más utilizadas en la actualidad.

Es un lenguaje de programación interpretado que se utiliza a través de la línea de comando aunque tiene la opción de ejecutar scripts. Las capacidades de R están extendidas ampliamente mediante la utilización de paquetes creados por usuarios tanto en R como en Java, C o Fortan. Existen más de 5.800 paquetes con 120.000 funciones (en junio 2014) en el almacén de paquetes de R (Comprehensive R Archive Network- CRAN).

Otra de las fortalezas de R es la capacidad de crear gráficos y símbolos matemáticos con la suficiente resolución para poder ser publicados. Son estos gráficos, de gran calidad, los que se utilizarán como resultado de la última fase de generación del conocimiento de este proyecto.

3.6 Hadoop

MapReduce [12] es una propuesta de programación funcional basada en los principios de administración de los datos, la abstracción de la paralelización/sincronización y la tolerancia a fallos. El modelo consiste en la utilización de dos primitivas: Map y Reduce. La entrada para el proceso Map es una lista de parejas (*key1*, *value1*). Los datos intermedios se agrupan en la forma (*key2*, *list(values2)*). Para cada key2 se ejecuta un Reduce con la lista de valores que produce cero o más resultados agregados. En la práctica se ha implementado en diversos productos de los que podemos destacar: Twister [17], LEMO-MR [13] o Hadoop [26]. Es esta última la de mayor renombre debido al apoyo que tienen de grandes corporaciones y de la comunidad de desarrolladores.

En sus inicios Hadoop fue creado por Doug Cutting, ingeniero de Yahoo, como un software de indexación de páginas web denominado Nutch. Después de la publicación de GFS (Google File System, GooFS o GoogleFS) [21] en octubre 2003 y MapReduce [12] en diciembre de 2004, recibió el nuevo nombre tras la incorporación de las nuevas funcionalidades y la adquisición de capacidades de escalabilidad. Ya en 2008, Yahoo se benefició de la potencia de Hadoop como herramienta para la búsqueda de webs con la utilización de aproximadamente 10.000 cores [35].

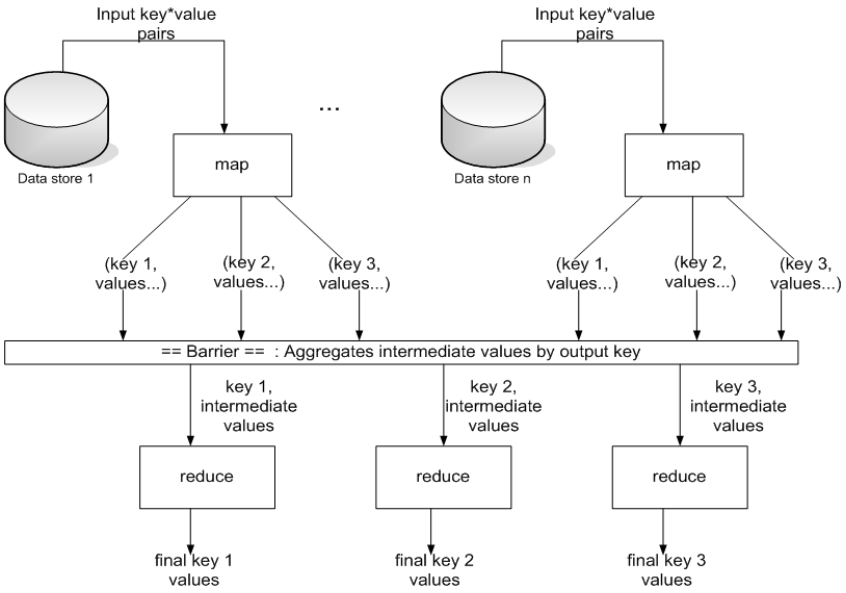

Figura 14 - Fases de una ejecución mediante la programación MapReduce

3.6.1 HDFS

HDFS (Hadoop Distributed File System) [27] es un sistema de ficheros distribuido diseñado para funcionar en entornos con alta tolerancia a fallos y desplegado en *commodity hardware*. En la actualidad es un subproyecto de Apache Hadoop.

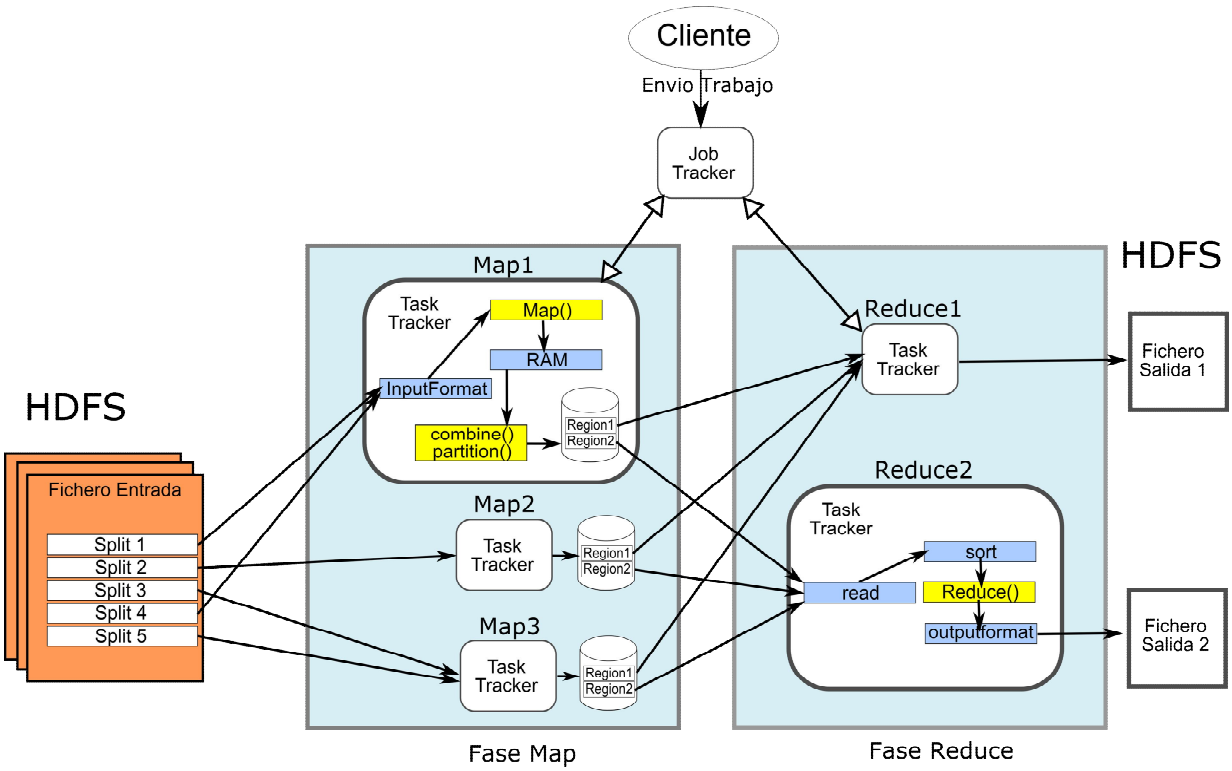

Figura 15 - Flujo de datos en Hadoop

Se diseñó con el objetivo de almacenar ficheros de gran volumen en un cluster de gran tamaño. Almacena los ficheros como una secuencia de bloques. Todos los bloques poseen el mismo tamaño excepto el último. Los bloques se replican con el fin de ser tolerantes a fallos. El tamaño del bloque y el factor de replicación son parámetros configurables de la instalación. En la instalación utilizada en el presente proyecto el bloque es de 128 MB y el factor de replicación es 2. Los ficheros son escritos una única vez y por un solo escritor.

Es el componente de ejecución (ver *JobTracker* en "Figura 15 - Flujo de datos en Hadoop") de los programas en Hadoop el que intentará ejecutar el programa en la localización donde se encuentre el bloque deseado en vez de desplazar el dato al lugar donde se debe procesar. Esta característica de aproximar el cómputo a los datos en lugar de los datos al cómputo es una de las principales características aportadas por Hadoop.

El espacio de nombres (*NameSpace*) es una jerarquía de ficheros y directorios. Los ficheros y directorios están representados en un servidor denominado *NameNode*. Los Inode representan los ficheros con atributos como pueden ser permisos, tiempos de acceso o cuotas de utilización del disco. Es función del *NameNode* mantener el árbol del espacio de nombres y el mapeo de bloques existentes en los *DataNodes*. Se puede configurar un *NameNode* secundario con el fin de almacenar la información replicada del primario.

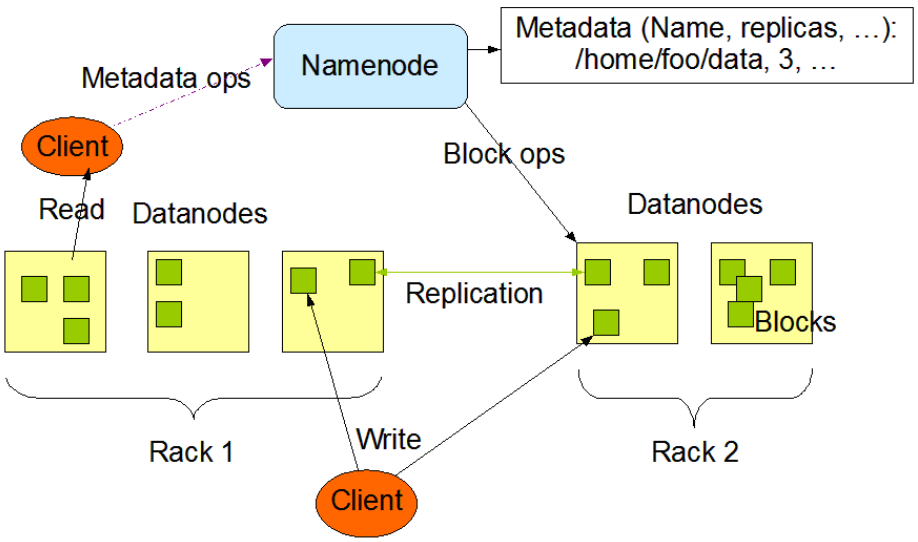

Figura 16 - Arquitectura HDFS [26]

4 Codificación y ejecución de los tratamientos

Todos los procesos anteriormente descritos se han implementado en tres grandes bloques:

- **Bloque 1 - Captura**, que incluye la fase de captura, adecuación y normalización de las emisiones.
- **Bloque 2 - Reconocimiento**, que tiene dos fases: diarización y reconocimiento vocal.
- **Bloque 3 - Generación del conocimiento**. Se ha diseñado como procesos independientes de generación de información procesable bajo demanda.

Como se ha comentado anteriormente el proceso de captura se ha independizado para componer una muestra con la suficiente entidad para realizar las pruebas del sistema que se desea.

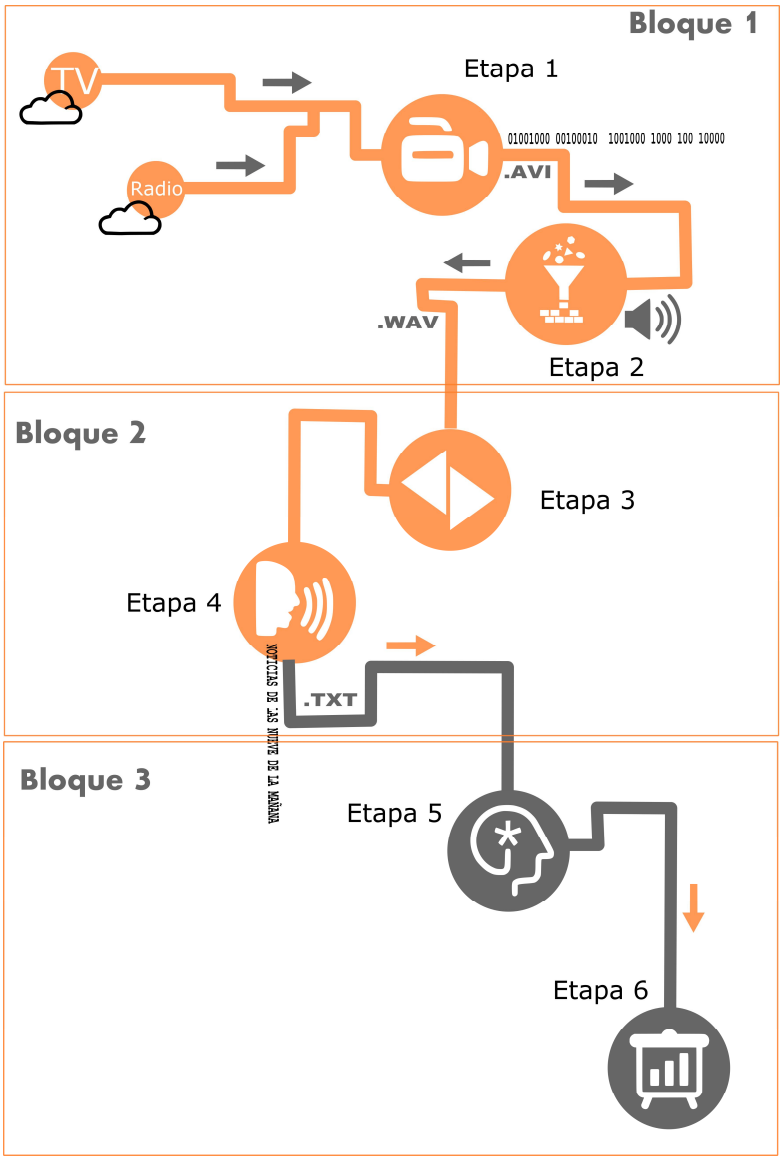

Figura 17 - Separación en bloques de implementación

4.1 Workflow Administrator: Apache Oozie

Con vistas a poder enlazar todos los procesos de la solución, se hace necesaria la utilización de un gestor de workflow adaptado al ecosistema de Hadoop. Entre las posibles opciones barajadas podemos encontrar Luigi [42], Azkaban [4] y Oozie [48]. La primera herramienta no soporta procesos MapReduce y entre las dos últimas, desarrolladas ambas en Java, se ha seleccionado la última por encontrarse en un estadio de desarrollo superior y permitir el lanzamiento de desarrollos en Pig, Hive, Sqoop y MapReduce. La elección de Oozie también ha estado marcada por su disponibilidad en el entorno de desarrollo de la distribución IBM InfoSphere BigInsights.

Apache Oozie, o simplemente Oozie, es un proyecto de código abierto de la fundación Apache que facilita la definición de los flujos de trabajos y la coordinación entre procesos. Un flujo de trabajo de Oozie se define como un DAG (*Directed Acyclic Graph*) ya que no pueden existir ciclos en el grafo (solo existe un punto de entrada y otro de salida). Aunque se le puede reprochar su complejidad por la utilización de ficheros de configuración XML, o que los procesos se ejecuten como trabajos Map en el cluster, lo cierto es que se encuentra perfectamente integrado dentro del portal de IBM BigInsights por lo que no ha sido necesario activar la consola web de la aplicación para controlar su ejecución ni el visor de logs de las aplicaciones ejecutadas.

Existe tres formas de lanzan los procesos de Oozie:

- Como **Job Bundle**. Abstracción realizada por la aplicación para manejar varias tareas Coordinator.
- Como **Job Coordinator**. Permite la ejecución de Oozie workflows de forma interdependiente, basados en periodos de tiempo o con dependencias de datos.
- Como **Job Workflow**. Son los flujos de trabajo que definen como un DAG.

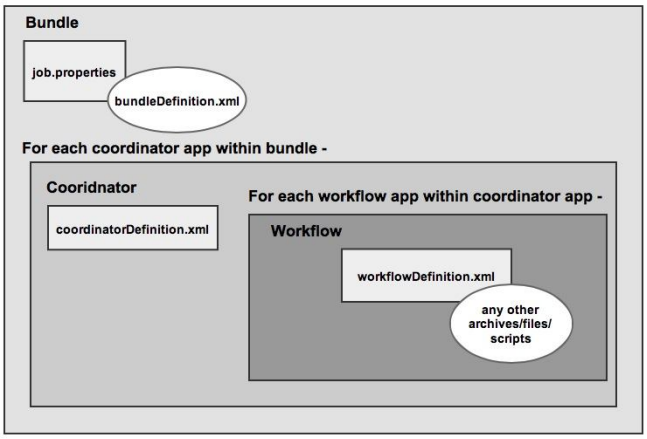

Figura 18 - Jerarquía de trabajos Oozie [29]

Para su lanzamiento desde la línea de comandos es necesario disponer de un fichero *.properties* que hace referencia al Bundle, Coordinator o Workflow que se desea ejecutar. Ver el "Anexo V - Ficheros Oozie" para tener más detalles acerca de los ficheros utilizados en el desarrollo.

4.2 Fases en la implementación

Como se ha comentado previamente la implementación se ha realizado ampliando los bloques que se han definido en las fases iniciales. De esta forma los bloques 2 y 3 se han subdividido en tareas como muestra el esquema de la "Figura 19 - Esquema de tratamiento completo":

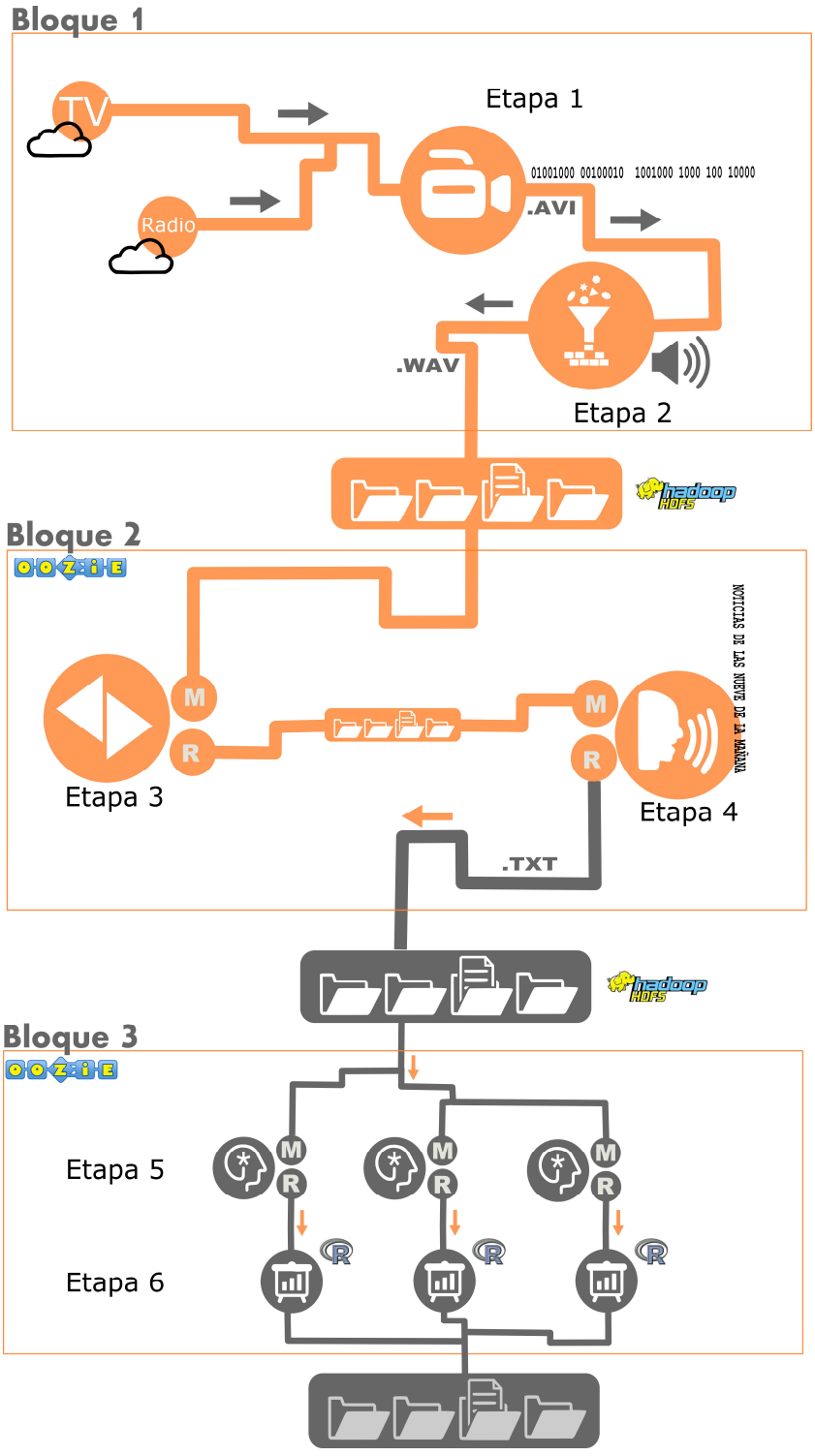

Figura 19 - Esquema de tratamiento completo

4.2.1 Captura

En esta primera fase se han incluido dos etapas del diseño original, la etapa de captura (etapa 1) y la etapa de adecuación y normalización de los ficheros (etapa 2).

La captura de los vídeos se ha realizado por medio de un script bash que utiliza dos paquetes de tratamiento de audio y vídeo. Este proceso de captura se ha ejecutado en una máquina externa al cluster Hadoop. En el caso de tratarse de vídeo (TV) se ha utilizado la aplicación *ffmpeg* para la captura mientras que para la captura de audio (radio) se ha utilizado *mplayer*. Todo el proceso de captura de las emisoras, hasta un máximo de 10, se ha realizado de forma paralela en una única máquina sin que suponga una sobrecarga en su funcionamiento normal.

Después del proceso de captura, se efectua una separación del vídeo y del audio, si es necesaria, para poder adecuar el audio al estándar fijado y que es necesario para las tareas posteriores. De esta forma se almacena un fichero por emisora y periodo en formato wav (Codec: Uncompressed 16 bit PCM audio, mono y frecuencia de muestreo 16.000 Hz).

Posteriormente se genera un único fichero con formato *sequence file* para el periodo capturado utilizando la API de Hadoop. Para realizar esta tarea se ha utilizado una aplicación Java que permite unir todos los ficheros capturados y convertidos a un formato de almacenamiento binario. Este formato de almacenamiento key/value, en nuestro caso, serán el nombre de la emisora (key) y la información binaria (value) del fichero wav que se ha generado previamente.

El nombre del fichero indicará la fecha de la captura con el formato

`NombreCadena_Año_Mes_Día_Hora_Minuto.sf`

Donde:

- `NombreCadena` muestra el nombre de la emisora capturada (no debe contener el carácter "_" ya que constituye el carácter de separación entre campos del formato).
- `Año`. Año de la captura de la emisión.
- `Mes`. Mes de la captura de la emisión.
- `Día`. Día de la captura de la emisión.
- `Hora`. Hora de captura de la emisión.
- `Minuto`. Minutos del inicio de la captura puede ser 00, 15, 30 ó 45.

Puede parecer duplicado el tener que almacenar la información en el nombre del fichero y también en el nombre del directorio que se va a utilizar. En efecto, el nombre que le ponemos al fichero, en esta fase, sirve para poder ubicarlo en el directorio correcto del sistema de archivos HDFS una vez transferido al cluster.

En el Anexo I - Ejemplo de implementación para el bloque 1" está disponible un resumen del script utilizado para capturar un grupo de emisoras reducido.

4.2.2 Reconocimiento

Una vez obtenidos los ficheros de audio con formato normalizado y almacenada la información resultante en la estructura de directorios definida, es el momento de realizar la transformación de los ficheros de sonido a texto. Este proceso se ha controlado por un job Coordinator de Oozie con dependencia de datos (ver en "Anexo V - Ficheros Oozie" los ficheros *newsASR.properties*, *newsASRCoordinator.xml*) denominado en la documentación de Oozie como "*Triggering Coordinator Jobs when Data file is Available*". Esto es, el job aunque se pone en funcionamiento no arranca los trabajos MapReduce hasta que el fichero no se encuentra en su directorio. De esta forma podemos lanzar el proceso que procesará los cuatro ficheros de una hora pero estos se esperarán a la finalización de la captura para ponerse en marcha. También se puede fijar el tiempo límite de espera de este tratamiento, de manera que si el periodo es superado, el proceso se muestre con estado TIMEOUT.

El proceso de Oozie es capaz de manejar toda la secuencia de procesos que a continuación se describen para lograr la generación un fichero de texto plano con la estructura key/value adecuada. La base de este proceso es la ejecución de dos jobs constituidos cada uno por varios procesos Map y un único Reduce.

- Proceso de la Etapa 3 (ver "Anexo III - MapReduce Etapa 3"). Diarización de los ficheros y almacenamiento en un *sequence file.*
- Proceso de la Etapa 4 (ver "Anexo IV - MapReduce Etapa 4"). Reconocimiento vocal.

La transferencia de información entre la etapa 3 y la etapa 4 se realiza mediante un fichero único, con formato *sequence file*, que es almacenado en HDFS en el directorio output del proceso correspondiente a la etapa 3. De la misma forma ocurre con el resultado del bloque 2 completo, que es el resultado de la etapa 4 pero en este caso se almacena un fichero de tipo texto. El código de la ejecución en la línea de comandos es:

```
biadmin@bivm:/mnt/data/newsASR> oozie job -config ./oozie/newsASR.properties -run
job: 0000329-150116232847899-oozie-oozi-C
```

Que tiene como resultado la ejecución de varios trabajos.

```
biadmin@bivm:/mnt/data/newsASR> oozie job -info 0000329-150116232847899-oozie-oozi-C
Job ID : 0000329-150116232847899-oozie-oozi-C
------------------------------------------------------------------------------------------
Job Name    : newsASR
App Path    : hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/newsASRcoordinator.xml
Status      : RUNNING
Start Time  : 2014-12-21 12:00 GMT
End Time    : 2014-12-21 14:00 GMT
Pause Time  : -
Concurrency : 1
% Complete : 50,00%
------------------------------------------------------------------------------------------
ID                                    Status    Ext ID                          Err Code  Created
Nominal Time
0000329-150116232847899-oozie-oozi-C@1     KILLED    0000330-150116232847899-oozie-oozi-W -         2015-
01-19 16:06 GMT 2014-12-21 12:00 GMT
------------------------------------------------------------------------------------------
0000329-150116232847899-oozie-oozi-C@2     SUBMITTED - - 2015-01-19 16:06 GMT 2014-12-21 12:15 GMT
------------------------------------------------------------------------------------------
0000329-150116232847899-oozie-oozi-C@3     READY     - - 2015-01-19 16:06 GMT 2014-12-21 12:30 GMT
------------------------------------------------------------------------------------------
0000329-150116232847899-oozie-oozi-C@4     WAITING   - - 2015-01-19 16:06 GMT 2014-12-21 12:45 GMT
```

Es importante observar el *Nominal Time*, en la última columna, que concuerda con el fichero que se está procesando independientemente del momento de la ejecución del proceso.

4.2.3 Generación de conocimiento

La generación de conocimiento se basa en la ejecución de varios procesos secuenciales independientes que tienen como resultado la realización de un gráfico con información de los datos tratados. Cada uno de los procesos diseñados amplia el volumen de tratamiento de los datos, de esta forma el primer proceso extrae datos de un espacio de tiempo reducido, aproximadamente de un par de horas como máximo. El segundo evoluciona hasta poder tratar información de un par de años y el tercero tiene un ámbito de aplicación que se extiende a varios años. Esto implica que cada uno de los proceso tiene un volumen de información creciente.

Los procesos diseñados son:

- **WorkKnow1**.- Este proceso tiene como fin la creación de una nube de palabras con la información de los periodos de tiempo elegidos. El volumen de datos que trata es pequeño ya que existe una limitación intrínseca del periodo de tiempo que se desea tratar.

Figura 20 - Resultado ejecución WorkKnow1

El proceso se compone de tres fases:

- Obtención de la lista de ficheros a tratar. Se genera una lista de ficheros correspondientes a los resultados de las capturas entre dos fechas indicadas. Esta lista obtenida con los nombres de los ficheros a tratar se le proporciona a la siguiente etapa.
- Procesamiento de los ficheros. Mediante un proceso MapReduce se realizarán dos tareas. La primera será la eliminación del grupo de palabras que no poseen significado para nuestro tratamiento (preposiciones, pronombres, artículos, etc.) comúnmente conocidas como *stopwords*. Cada idioma tiene su propia lista de

stopwords y en nuestro caso se utiliza una lista propia del castellano. La segunda tarea es la sumarización o agregación de las apariciones de las palabras de los reconocimientos seleccionados en la fase anterior. El resultado será la lista de palabras a representar con su número de apariciones en el periodo.

- Generación de los gráficos. Con la utilización de R se generará un grupo de gráficos (nubes de palabras) que mostrarán el contenido de los datos tratados.

  En la "Figura 20 - Resultado ejecución WorkKnow1" se muestra la captura de la generación de una nube de palabras utilizando un juego de pruebas basado en el corpus del modelo de reconocimiento utilizado en el proceso de entrenamiento.

- **WorkKnow2**.- En este segundo proceso de generación del conocimiento se pretende detectar el aumento de frecuencia de la aparición de ciertas palabras. Utilizando un grupo de palabras introducido en un fichero se tratará de encontrar sus apariciones en el periodo de tiempo que se indique.

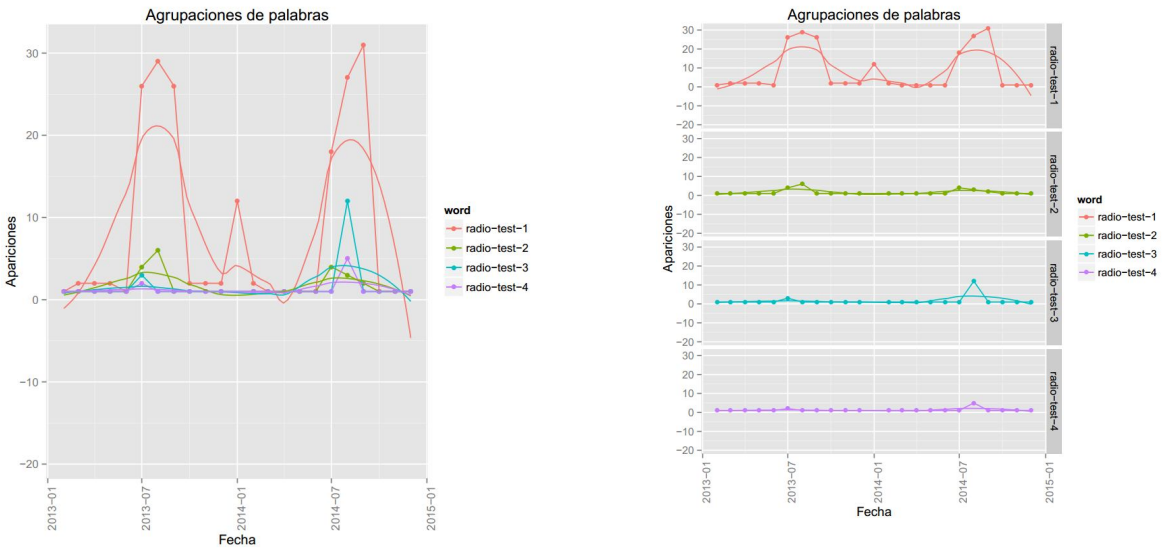

Figura 21 - Resultado ejecución WorkKnow2

Las tres fases que lo componen son:

- Obtención de los ficheros a tratar. De similar forma que en el workflow1 se obtiene una lista de ficheros a tratar, en este proceso se obtiene una lista de ficheros pero más extensa.
- Filtro del contenido de los datos de la lista especificada mediante MapReduce. Las palabras que se seleccionarán son aquellas que sean iguales a las indicadas dentro de un fichero. De la misma forma que el proceso de generación de conocimiento anterior, también se eliminan las *stopwords* que son encontradas.
- Representación en un gráfico. Se ha realizado la presentación de los resultados mediante un gráfico de líneas que permite mostrar la frecuencia de aparición mensual del grupo de palabras buscada. Todo ello se presenta sobre un eje de tiempo.

  Para la obtención de la imagen "Figura 21 - Resultado ejecución WorkKnow2" se ha utilizado un conjunto de datos que reproduce los reconocimientos realizados

durante dos años. Para ello, se ha copiado tantas veces como ha sido necesario el resultado de un reconocimiento y se ha alterado la frecuencia de aparición de algunas palabras. En el caso que se muestra en la imagen se alteraron las apariciones de las palabras "azul" y "ojeda" con el fin de mostrar una mayor aparición en los momentos deseados. El gráfico muestra, durante un periodo extenso de tiempo (dos años), la aparición de singularidades en periodos de tiempo concretos que son exactamente los que fueron preparados en la prueba. Todo ello se presenta de dos formas: agrupada (izquierda) o individualizada por emisoras (derecha). Todos los gráficos tienen sobreimpreso el resultado de la función de *smooth* que procura que el gráfico no sea tan dependiente del ruido o de fenómenos eventuales.

- **WorkKnow3**.- En este proceso se pretende reconocer la aparición de ciertas palabras durante un periodo elevado de tiempo. Se ha buscado un modo de representación más adecuado para este tipo de información como es el Mapa de Calor. En el caso que se presenta, la extensión de los ficheros utilizados es de 10 años. La secuencia de ejecución se ha compuesto de solo dos subprocesos, un primer subproceso MapReduce y un segundo proceso de representación gráfica con R. Para hacer posible esto, los datos de las fechas seleccionadas deben ser proporcionadas al proceso MapReduce para que realice la selección de los ficheros que se encuentren en el periodo. Este mismo subproceso elimina las *stopwords* y selecciona las palabras de interés. En la sección "5 - Mediciones en los procesos de tratamiento y representación de los datos" se analiza más profundamente el rendimiento de los subprocesos y las decisiones de diseño que se han realizado.
-

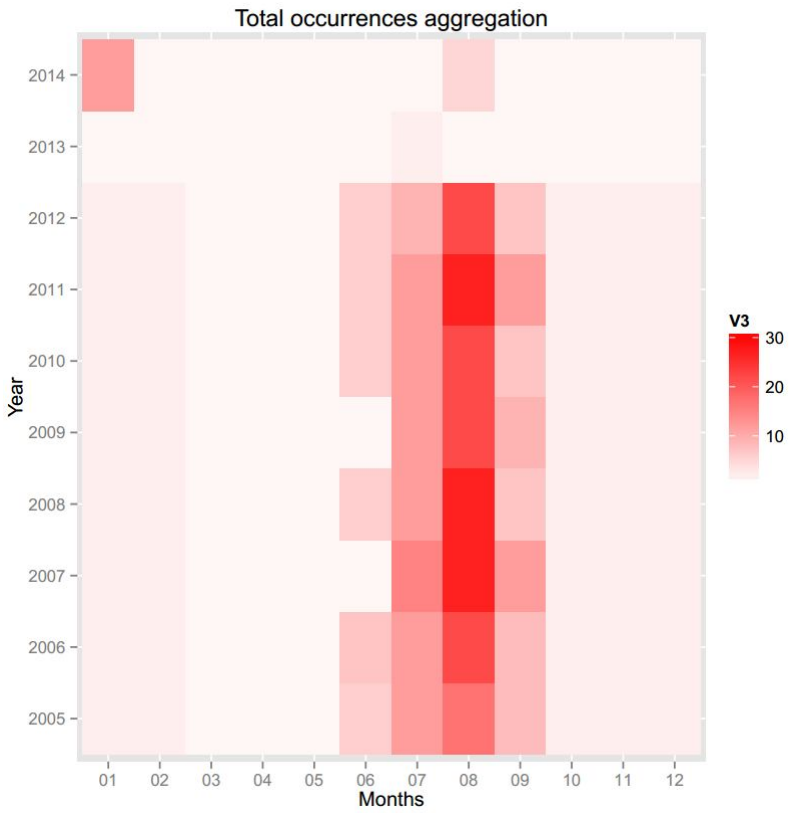

Figura 22 - Resultado ejecución WorkKnow3

  - De la misma forma que se hace en el WorkKnow2 se seleccionan únicamente aquellas palabras que aparecen dentro de un fichero y se eliminan las *stopwords*,

todo ello mediante un proceso MapReduce. Este proceso además realiza la selección entre las fechas que se desea en función de los parámetros que se le han proporcionado.

- Generación del gráfico. Se ha realizado la presentación de los resultados mediante un mapa de calor que resulta más adecuado para la extensión temporal de los datos que se seleccionan.

En la "Figura 22 - Resultado ejecución WorkKnow3" se muestra la aparición de las mismas dos palabras que se han utilizado anteriormente (“azul” y “ojeda”) pero en un juego de prueba más amplio (10 años).

La ejecución de cualquiera de los tres procesos de generación del conocimiento se puede realizar en la línea de comandos. La modificación del fichero *properties* (ver "4.1 - Workflow Administrator: Apache Oozie") facilita la ejecución de cualquiera de los tres procesos.

```
biadmin@bivm:/mnt/data/newsASR> oozie job -config ./oozie/createKnow2.properties -run
job: 0000334-150116232847899-oozie-oozi-C
```

Se puede observar el proceso en ejecución

```
biadmin@bivm:/mnt/data/newsASR> oozie job -info  0000334-150116232847899-oozie-oozi-C
Job ID : 0000334-150116232847899-oozie-oozi-C
------------------------------------------------------------------------------------------------------------------------
Job Name    : createKnow2
App Path    : hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/createKnow2Coordinator.xml
Status      : RUNNING
Start Time  : 2014-12-21 12:00 GMT
End Time    : 2014-12-21 12:01 GMT
Pause Time  : -
Concurrency : 1
% Complete : 100,00%
------------------------------------------------------------------------------------------------------------------------
ID                                        Status    Ext ID                              Err Code  Created
Nominal Time
0000334-150116232847899-oozie-oozi-C@1     RUNNING   0000335-150116232847899-oozie-oozi-W -         2015-01-
19 16:16 GMT 2014-12-21 12:00 GMT
------------------------------------------------------------------------------------------------------------------------
```

Los ficheros necesarios para la ejecución se encuentran en el "Anexo V - Ficheros Oozie". El resultado se ha almacenado en el directorio temporal (temp) donde se pueden ver los ficheros generados con formato pdf (generate.pdf)

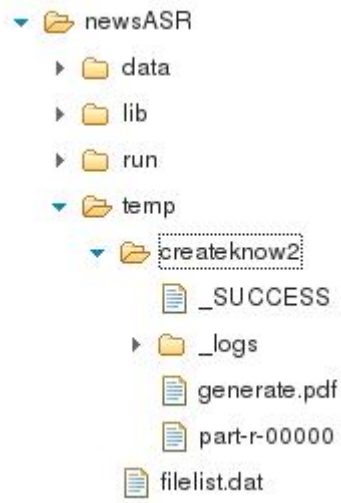

Figura 23 - Captura del sistema de ficheros HDFS después de la generación del WorkKnow2

5 Mediciones en los procesos de tratamiento y representación de los datos

Después de la implementación de todos los procesos, disponer de un código ejecutable y verificar su correcto funcionamiento, se hace necesario realizar una valoración del desempeño del sistema de una forma cuantificable en base al conjunto de pruebas de Voxforge utilizado. Las medidas del WER (Fórmula 1) y Accuracy (Fórmula 2) son muy dependientes de los procesos de reconocimiento y por lo tanto hablar de ellos no es una cuestión que esté dentro del ámbito del presente proyecto. Sin embargo, para conocer un poco más el sistema de reconocimiento que hemos utilizado podemos indicar que tiene un WER de 18% y Accuracy de 84%. Los cálculos se han realizado con 4 segmentos del juego de voces utilizado en el entrenamiento del modelo (CarlosEspinoAngulo-20140525-Vip es-0019.wav, es-0020.wav, es-0021.wav y es-0022.wav [55]) usando un total de 71 palabras (I=2, D=9, S=2, N=71) y un total de 36 segundos de duración. El reconocimiento ha terminado en 54 segundos. Por lo tanto, se obtiene un sistema 1.5xRT (Fórmula 3) con un solo proceso en funcionamiento.

El estudio se centrará más en las posibilidades de paralelización del sistema, las mediciones de las diferentes partes de los procesos, su variación con la inclusión de nuevos nodos y la observación del tamaño de los ficheros tratados o generados así como otros parámetros que limitan el nivel de desempeño.

5.1 Análisis del comportamiento HTC del sistema.

La ley de Amdahl nos marcará el límite inferior de ejecución ("*La parte secuencial de un programa determina una cota inferior para el tiempo de ejecución, aún cuando se utilicen al máximo técnicas de paralelismo*"). En nuestro caso podemos considerar la ejecución de la fase 2 de reconocimiento como:

$$T_{total} = \frac{TMap_{diariza}}{m} + TRed_{diariza} + \frac{TMap_{asr}}{n} + TRed_{asr}$$

Fórmula 4

Donde:

- T_{total} es el tiempo que tardará el bloque 2 en ejecutarse.
- $TMap_{diariza}$ es el tiempo del proceso Map de la fase de diarización (Etapa 4).
- $TRed_{diariza}$ es el tiempo del proceso Reduce de la fase de diarización (Etapa 4).
- $TMap_{asr}$ es el tiempo del proceso Map de la fase de reconocimiento (Etapa 5).
- $TRed_{asr}$ es el tiempo de proceso Reduce de la fase de reconocimiento (Etapa 5).
- m es el número de nodos que se utilizan en el proceso de diarización.
- n es el número de nodos que se utilizan en el proceso de ASR.

por tanto nos llevaría a que $TRed_{diariza}$ + $TRed_{asr}$ es el tiempo mínimo teórico que podemos llegar a obtener para la realización de la tarea.

Es fundamental conocer el sistema que utiliza Hadoop para la generación de procesos y el modo como se reparten los datos entre los diferentes procesos. Ajustar el tamaño del bloque procesado

puede aproximarnos a un conocimiento mayor del tratamiento que estamos realizando. Los *sequence files* se procesan de la misma forma que un fichero de tipo texto por líneas y/o bloques de líneas. En los *sequence files* se trata cada key/value como si fueran un único registro que se procesa dependiendo de los bloques. Es necesario, por lo tanto, realizar un estudio un poco más pormenorizado de la dimensiones de los ficheros/tamaño del bloque y la paralelización que esto produce.

El tamaño del bloque utilizado en Hadoop caracteriza el total de procesos paralelos que se ejecutan en el cluster. Los datos adquiridos en este apartado han sido obtenidos a partir de un fichero *sequence file* de pruebas con una duración de 15 minutos que pretende ser la representación de la captura de 10 emisoras y que ha sido construido a partir del Corpus de Voxforge [55]. El tamaño total del fichero generado ha llegado a los 464 MB.

Existen dos métodos para indicar el bloque de datos que tratará un proceso en Hadoop. La primera opción, bloque físico, la proporcionan los propios ficheros de Hadoop ya que se permite generar ficheros con tamaño de bloque determinado o copiar al sistema de ficheros Hadoop (HDFS) documentos con un tamaño de bloque diferente al indicado por defecto. Para ello se puede utilizar el comando:

```
hadoop fs -D dfs.block.size=134217728 -put  from_localfile to_hdfs_file
```

o bien en el ficheros de configuración del workflow (workflow.xml) para generar fichero con tamaño de bloque definido:

```
<property>
    <name>dfs.block.size</name>
    <value>${blockSize}</value>
</property>
```

Podemos conocer los bloques que se han utilizado en el almacenamiento de un fichero, su situación y las replicas que tiene utilizando el comando

```
hadoop fsck -blocks -files -locations sequence_file.sf
```

La segunda opción, bloque virtual, se basa en repartir un mismo bloque físico escrito en un *DataNode* entre varios procesos. Para esto hay que utilizar la variable *mapred.max.split.size* que permite modificar en un proceso el valor de split que tiene el sistema y que por defecto es igual al del bloque físico por defecto del sistema.

```
<property>
    <name>mapred.max.split.size</name>
    <value>${SplitSize}</value>
</property>
```

El segundo método, bloque virtual, es más adaptable a las necesidades de procesamiento ya que se puede indicar en el inicio de la ejecución del proceso mientras el primero, el bloque físico, debe indicarse en el momento de la grabación del fichero. El segundo método es más adaptable a las características de la infraestructura y/o el tratamiento que se desee realizar mientras que el primero se debe conocer en el momento de generación del fichero siendo por tanto menos flexible. Para modificar el tamaño del bloque físico hay que volver a copiar el fichero completo con el tamaño de bloque deseado.

Las pruebas realizadas para la evaluación de tiempo de respuesta con los dos sistemas de reparto de los datos en los procesos dan valores medios similares. Para realizar esta comparación se ha utilizado bloques físicos de 32 MB y por otro lado el bloque físico de 128MB (por defecto del sistema) con selección de bloque virtual de 32 MB. Se ha seleccionado el tamaño de 32 MB ya que proporciona un mayor número de bloques, como veremos con posterioridad, y por lo tanto un mayor paralelismo. En los dos casos, es evidente, se han ejecutado 57 procesos en la etapa 4 de reconocimiento vocal teniendo una media de tiempos próxima en los dos casos a las 3h 15 min (etapa etapa3 + etapa4). Por lo tanto en este punto, basandonos en nuestra infraestructura de prueba (existe una descripción detalla en la sección 5.2), y teniendo solo en cuenta la velocidad de ejecución, no se puede concluir que un método sea mejor que el otro.

Podemos analizar un poco más como se ha producido la ejecución de los procesos Hadoop en función de la división del bloque que se ha realizado. En el primer caso, utilizando el *mapred.max.splits.size* de 32MB, en el que la división del bloque es virtual, y el bloque físico es de 128MB, vemos que se intenta no desplazar los grandes bloques de los ficheros y se reservan los bloques virtuales para su futura ejecución en los nodos (los dos primeros procesos se ejecutan en el nodo 2 – jobs 1 y 2 - y los otros dos en el nodo 1 – jobs 12 y 13-). Se puede concluir que hace una evaluación de la ejecución futura.

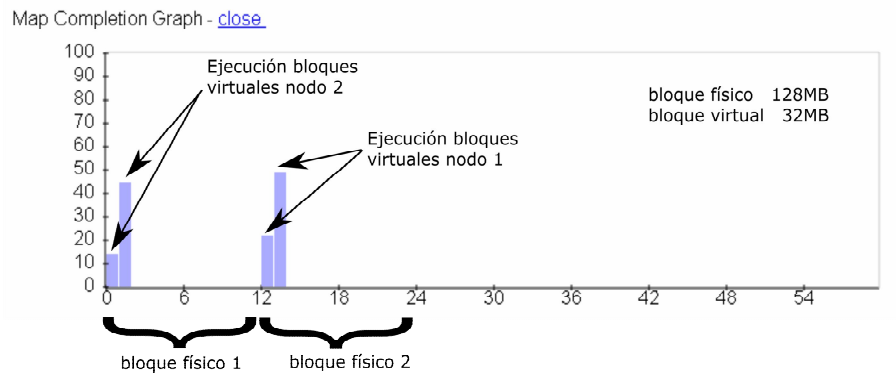

Figura 24 - Captura de la ejecución con bloque virtual (% ejecución/proceso)

En el segundo caso, se ejecuta con un fichero con tamaño de bloque físico de 32 MB y bloque virtual 32MB. El reparto de los bloques es más consecutivo que en el caso anterior. Aunque se sigue dando preferencia al lugar donde esta guardado el bloque para su ejecución, no se ha producido una reserva de partes de bloques.

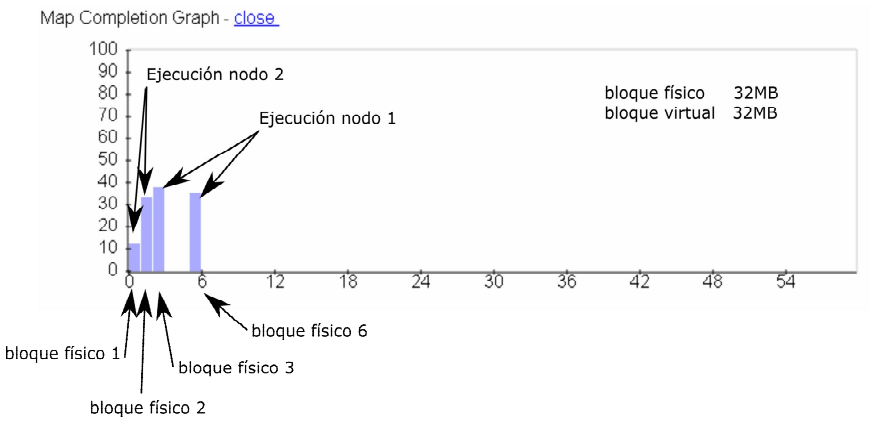

Figura 25 - Captura de la ejecución con bloque físico (%ejecución/proceso)

Habría que tener en cuenta también el número de copias que se poseen de cada bloque ya que Hadoop puede utilizar cualquiera de las copias disponibles en los nodos para realizar el tratamiento tanto del bloque físico como del virtual. Con el tamaño que tenemos de pruebas no se puede observar la utilización de las copias de bloques.

El número total de bloques que se obtiene modificando el parámetro de *block.size* (bloque físico) para el fichero de prueba se indica en la "Tabla 4 - Total de bloques de datos obtenidos en las etapas 3 y 4" en función de cuatro medidas básicas. El tamaño de 128MB es el utilizado por defecto en la instalación de prueba. El tamaño de un bloque en un sistema de ficheros ext2 o ext3 puede llegar a 8KB.

Proceso/Tamaño bloque	32MB	64MB	128MB	256MB
Etapa 3	15	8	4	2
Etapa 4	57	29	14	7

Tabla 4 - Total de bloques de datos obtenidos en las etapas 3 y 4

De esta tabla podemos extraer el máximo nivel de paralelismo alcanzable para el sistema. Los 57 bloques indican un máximo de 57 procesos ejecutándose en paralelo en la etapa 4, por lo tanto una vez llegados a este máximo la inclusión de nuevos nodos no obtendrá ninguna mejora en el resultado. Una posible mejora sería la combinación de los dos métodos. La realización de un bloque físico de 32MB y uno virtual de 16MB llevaría a la existencia de un número mayor de bloques pero no conseguiría nada que no se pudiera lograr con la indicación de un tamaño de bloque virtual de 16 MB o de bloque físico de 16MB de forma exclusiva.

El balanceo de la carga de trabajo es un factor relevante que nos da una idea del desempeño de aplicaciones paralelas. En el análisis que hemos realizado podemos entender que el balanceo de la carga no se produce entre los nodos existentes única y exclusivamente sino que depende de los bloques que tenemos pendientes de procesar, el balanceo será dependiente del menor valor entre el número de bloques de los ficheros y el número de nodos.

5.2 Información temporal y espacial de las fases

Para la obtención de tiempos se ha utilizado un pequeño cluster compuesto de tres máquinas virtuales desplegadas en la infraestructura de Cloud on-premise basada en OpenNebula del GRyCAP.

Nombre	Función	Características de nodo
bivm	NameNode, Scheduler, Catalog, Bigsql-server, Zookepper-client, DataNode, TaskTracker, SecondaryNameNode, JobTracker, Head-node	2 CPU QEMU GenuineIntel 2GHZ 4GB memoria principal DD 128GB
node1	DataNode, TaskTracker	2 CPU QEMU GenuineIntel 2GHZ 4GB memoria principal DD 71GB
node2	DataNode, TaskTracker	2 CPU QEMU GenuineIntel 2GHZ 4GB memoria principal DD 71GB

Tabla 5 - Composición del cluster Hadoop

La configuración de cada uno de los nodos permite la ejecución de 2 tareas Map y 2 tareas Reduce, excepto la máquina bivm que solo acepta realizar una única tarea Reduce. En total se dispone de capacidad para la realización de 4 tareas Map y 5 tareas Reduce (ver en la Figura 26 las columnas *MapTaskCapacity* y *ReduceTaskCapacity*). Se ha configurado de esta forma para no limitar el trabajo del cluster por la sobrecarga de la máquina bivm que contiene, además del *DataNode* y el *TaskTracker* que son los procesos propios de cualquier nodo de un cluster Hadoop, los procesos de infraestructura como pueden ser el *NameNode* y *JobTracker*. Las posibilidad de tener 5 tareas Reduce no limitará nunca la ejecución de las tareas de los procesos diseñados ya que solo se requieren un único Reduce para cada etapa MapReduce diseñada.

Cluster Summary (Heap Size is 71.06 MB/2.06 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	1055	3	0	0	0	0	4	5	3,00	0	0	0

Figura 26 - Captura del *Cluster Summary*

El volumen máximo de datos almacenable en HDFS es de 112 GB teniendo configurados 248 GB como espacio para todo el sistema de ficheros distribuido. Aunque el tamaño máximo de almacenamiento se ha variado en numerosas pruebas no se ha observado que sea un factor relevante en la ejecución temporal de los procesos.

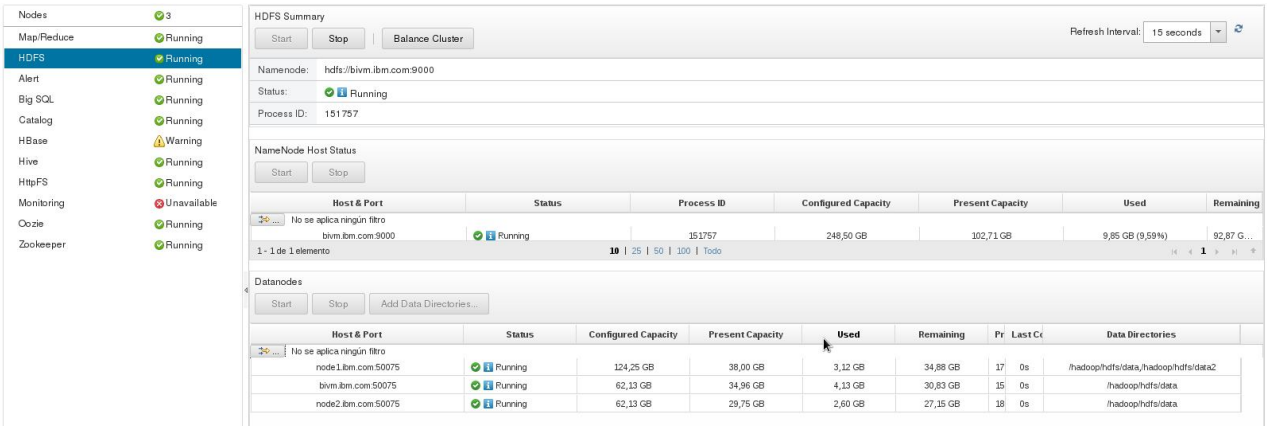

Figura 27 - Captura de la información de los nodos en la infraestructura de pruebas

La utilización de un tamaño de bloque pequeño supone una posibilidad de paralelización mayor del proceso pero implica una mayor sobrecarga en los procesos de transferencia de datos. Vamos a profundizar un poco más en los datos en función de tamaño del bloque.

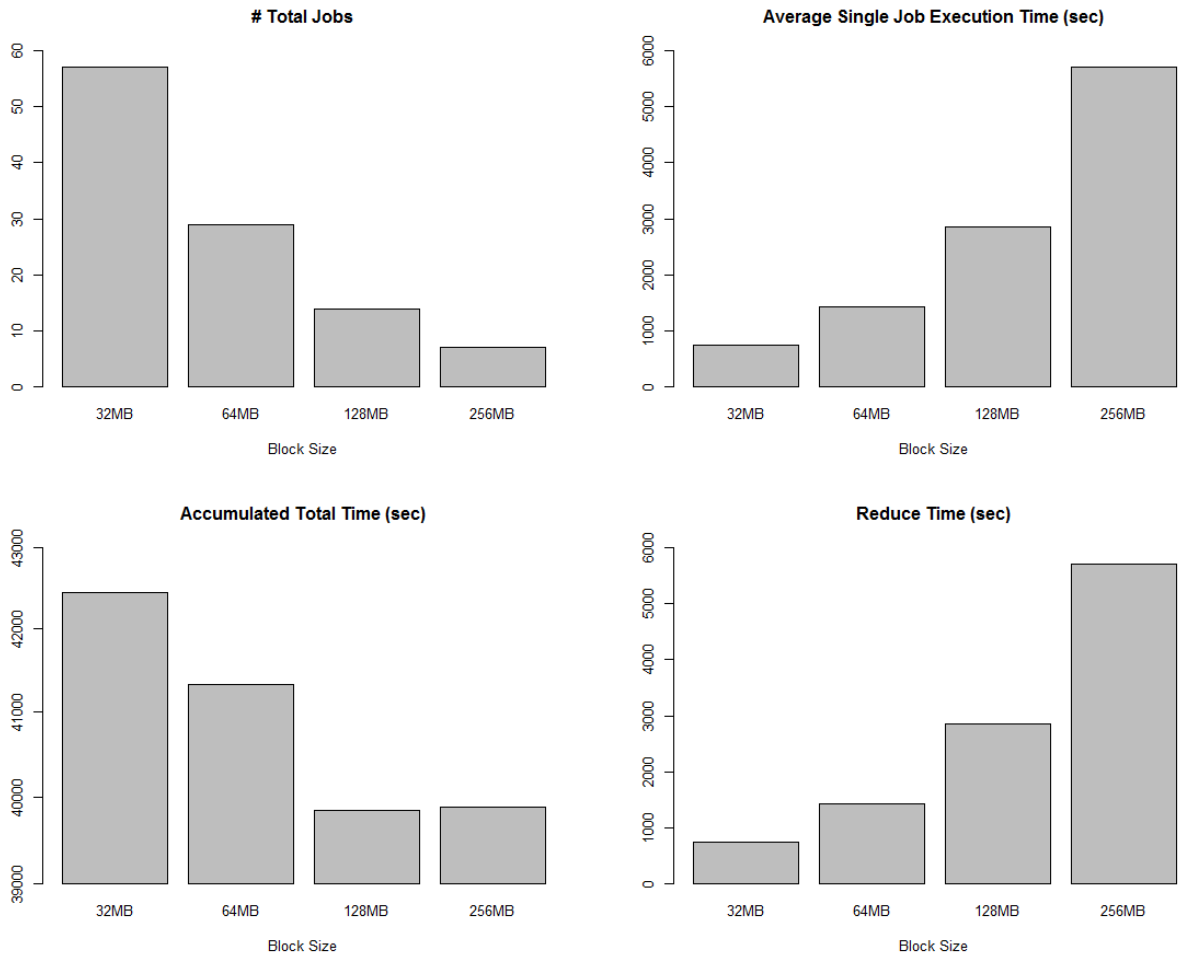

Figura 28 - Ejecuciones de la etapa 4 en función del tamaño del bloque

En la parte superior izquierda de la Figura 28 podemos ver de forma gráfica el total de trabajos que se han ejecutado en función del tamaño del bloque.

En el segundo gráfico (superior derecha) se ha realizado una media de los segundos que tarda en completarse uno de los trabajos Map, ya que el tamaño del bloque supone una mayor utilización temporal de los recursos para completar el trabajo.

	32 MB	**64 MB**	**128 MB**	**256 MB**
Total jobs	57	29	14	7

Tabla 6 - Total de jobs en función del tamaño del bloque (Etapa 4)

	32 MB	**64 MB**	**128 MB**	**256 MB**
Media jobs (seg)	744	1424	2.845	5.698
Total jobs (seg)	42.448	41.323	39.839	39.886
Reduce (seg)	4.982	5.053	5.350	5.652

Tabla 7 - Evaluación en función del tamaño del bloque (Etapa 4)

En el tercer gráfico (inferior izquierda) podemos ver el tiempo total que se ha utilizado para completar el mismo trabajo. Se puede observar que el tiempo total, que es la suma de todos los procesos que han conducido a la realización de la tareas, es mayor contra menor es el tamaño del bloque debido a la sobrecarga del tratamiento de los bloques a utilizar.

En el cuarto gráfico (inferior derecha) vemos un comportamiento menos predecible. El tiempo que tarda la ejecución del job Reduce, que es único por proceso, es mayor en los bloques de gran tamaño que en los pequeños. Suponemos siempre que la transferencia de pequeños bloques supone una sobrecarga en el sistema pero en el caso que nos ocupa cuesta más tratar los bloques de tamaño elevado que los de menor tamaños. El proceso Reduce recibe los datos de los procesos Map que se estén ejecutando. De esta forma con bloques de 32 MB y 57 porciones datos ejecutadas en 57 jobs independientes es más rápido que el tratamiento de 7 porciones para bloques de 256MB. No parece lógico que el tratamiento de la misma cantidad de datos cueste más al procesarlos en bloques grandes que en bloques pequeños. Por tanto, es necesaria una investigación más detallada sobre comportamiento menos predecible.

A continuación se muestra la duración estimada en segundos de las etapas 3 y 4 para la ejecución del Bloque 2 con un único proceso Map:

	Etapa 3		**Etapa 4**		**Total Bloque 2**
	Map	Reduce	Map	Reduce	
32 MB	2.051	377	42.448	4.982	49.858
64 MB	2.067	436	41.323	5.053	48.879
128 MB	1.988	558	39.839	5.350	47.735
256 MB	1.797	573	39.886	5.652	47.908

Tabla 8 - Estimación de tiempos totales en Bloque2 (#Map=1) (en seg)

Con el fin de conocer el tiempo que ha tardado en ejecutarse en la infraestructura de prueba, en la "Tabla 9 - Tiempo total ejecución del proceso Bloque2 en la infraestructura de pruebas" se puede observar el tiempo total de ejecución del bloque 2, en horas o segundos, lo que nos indica que supone una sobrecarga el realizar bloques de tamaño más pequeños.

Tamaño bloque	T_{ini}-$T_{fin\ (en\ seg)}$	T_{ini}-$T_{fin\ (en\ h)}$
32MB	11.481	3,2
64MB	12.087	3,3
128MB	12.445	3,4

Tabla 9 - Tiempo total ejecución del proceso Bloque2 en la infraestructura de pruebas

Desde este punto se analizarán individualmente cada una de los bloques de ejecución y se comentarán los posibles parámetros relacionados con los tiempos de ejecución que se han observado.

5.2.1 En el bloque de Captura

Debido a que se han definido periodos de tiempo fijos para la captura, el proceso de captura tiene invariablemente una duración de 15 min. Se ha considerado este un tiempo adecuado para las

capturas. La fase de captura no la podemos considerar como parte del algoritmo paralelo que se ejecuta en Hadoop aunque la ejecución de las diferentes partes se realiza en paralelo.

Durante este bloque se generan los *sequence file* que son necesarios para iniciar el tratamiento.

5.2.2 En el bloque de Reconocimiento

El bloque de ejecución del reconocimiento es una de las fracciones más importantes del proceso y la porción que permite una mayor paralelización. Teniendo en cuenta los datos que hemos obtenido en la infraestructura de prueba podemos llegar a conocer el comportamiento que obtendremos del sistema valorando ciertas medidas.

SpeedUp

El *SpeedUp* es una medida de la mejora del rendimiento de una aplicación al aumentar la cantidad de procesadores comparada con el rendimiento al utilizar un solo procesador. Podemos definir el *SpeedUp* absoluto como el cociente entre el tiempo del mejor algoritmo secuencial y el tiempo de la ejecución en N procesadores. Ya que en ocasiones no se dispone del mejor algoritmo secuencial se especifica el *SpeedUp* algorítmico, que se define como el cociente del tiempo de ejecución en un único procesador y la ejecución en paralelo con N procesadores [14]. Es este último el que hemos utilizado para realizar el gráfico "Figura 29 - Previsión de *SpeedUp* en función del número de nodos utilizados en la etapa 4" que muestra el *SpeedUp* con diferentes tamaños de bloque en el que se aprecia un *SpeedUp* supralineal en los cuatro casos para la etapa 4.

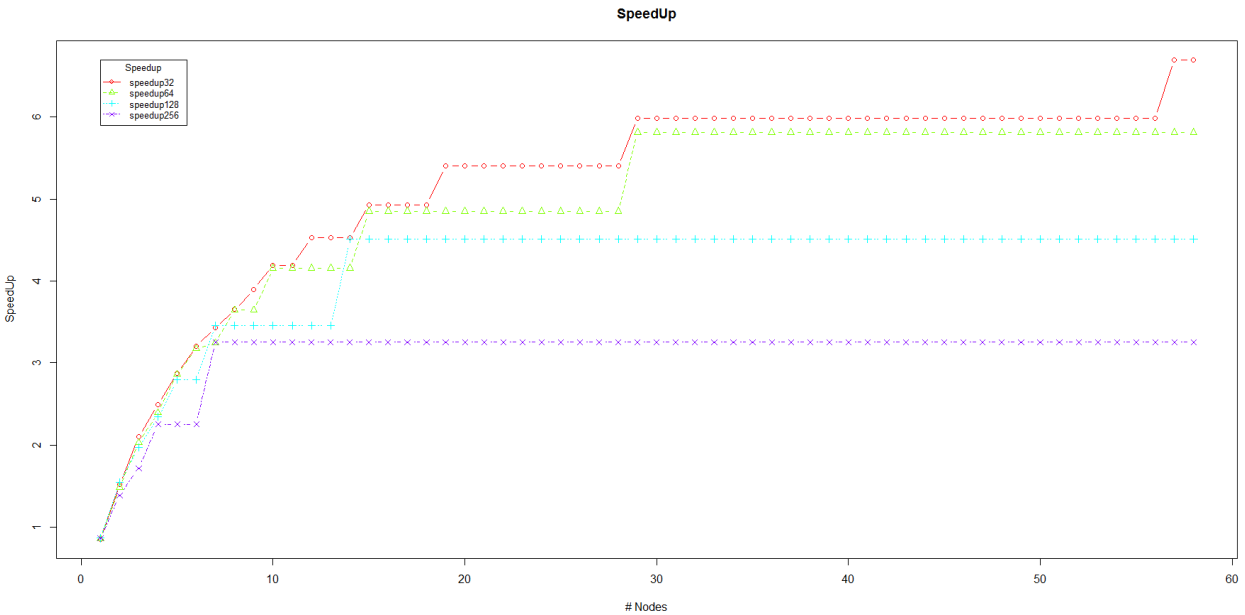

Figura 29 - Previsión de *SpeedUp* en función del número de nodos utilizados en la etapa 4

En el eje de abscisas (x) se representa la cantidad de nodos que se evaluan y en el eje de ordenadas (y) se representa la estimación del *SpeedUp* que se prevé. Para la obtención de esta estimación se realizó la ejecución real obteniéndose la duración media de cada uno de los bloques en la infraestructura de pruebas. Una vez obtenidos los tiempos medios de ejecución de un bloque se calculó el tiempo necesario para que esos mismos bloques se ejecutaran en el número de nodos indicado. En el caso de bloques grandes (256MB) vemos que desde 7 en adelante no se consigue

ningún incremento de la velocidad mientras que en el caso de bloques pequeños (32MB), se consiguen incrementos escalonados hasta llegar a los 57 nodos que ya no se consigue ningún incremento. Por tanto, si deseamos bloques de 128MB, la teoría indica que desde 14 nodos no se consigue ninguna mejora con la inclusión de nuevos nodos.

Eficiencia computacional

Otra medida que podemos obtener para conocer las bondades de nuestro sistemas es la eficiencia computacional, parámetro que se define como el cociente *SpeedUp*/número procesadores que nos traza la "Figura 30 - Previsión de eficiencia en función del número de nodos utilizados" también para la etapa 4. Como podemos ver el eje x representa el número de nodos y el eje y representa la eficiencia. La eficiencia es mejor cuanto más próxima a 1 este. Se muestran resultados en función del tamaño de bloque.

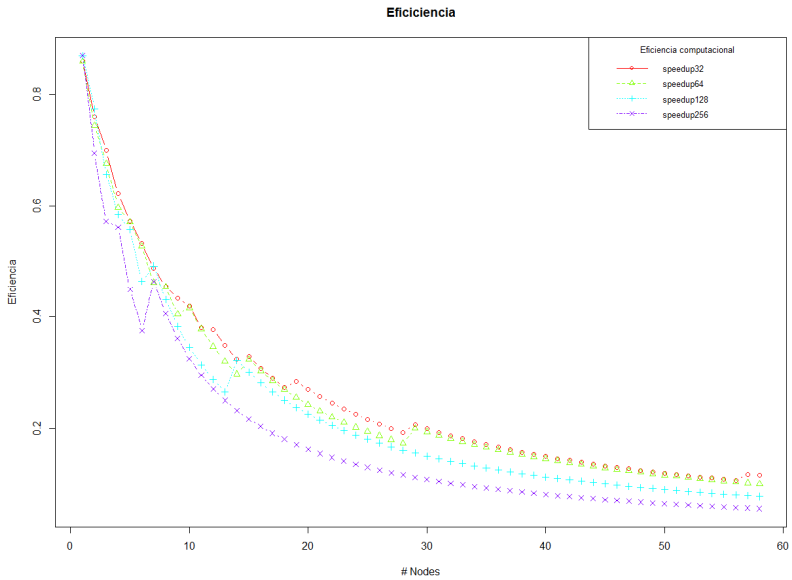

Figura 30 - Previsión de eficiencia en función del número de nodos utilizados

5.2.3 En el bloque de generación de conocimiento

En esta sección analizaremos los tiempos empleados en la ejecución de los tres procesos de generación del conocimiento (workKnow) que corresponden al bloque de ejecución 3.

La primera fase que se ejecuta es la obtención de la lista de los ficheros que se encuentran entre las fechas marcadas. Si tenemos en cuenta que un fichero con 10 registros correspondiente a 10 referencias de los ficheros de reconocimiento ocupa 480 Bytes (extraido del fichero en ejecución del workflow1 con tiempo de tratamiento de 2 horas y media), para conseguir un bloque 32MB (33.554.432 Bytes), el más pequeño que es capaz de generar paralelismo -con 2 bloques-, es necesario tener una lista correspondiente a 7.282 días para lograr un segundo proceso en funcionamiento. ((33.554.432 Bytes * 10/480 Bytes) ficheros necesarios / 4 ficheros por hora/24 horas del día = 7281.7 días). Esto significa que no se obtendrá ningún tipo de paralelismo en el tratamiento de los ficheros en los WorkKnow hasta que no hagamos estudios donde el número de ficheros incluya un número igual o superior a 20 años.

Teniendo en cuenta que el WorkKnow1 tiene una orientación reducida en el tiempo (podríamos aproximarlo a una ejecución en el espacio de tiempo de la última hora). El WorkKnow2 tiene una orientación más amplia llegando a los 2 años y por último el WorkKnow3 tiene una previsión de varios años. La ejecución de los dos primeros tendrá el resultado que se muestra en la Tabla 10 - Ejecuciones de la generación de conocimiento" mientras que la extrapolación, ya que son procesos similares, nos permitirá el cálculo de tiempo del tercero.

	Ámbito de ejecución	Tiempo ejecución
WorkKnow1	1 hora	30 sec. (real)
WorkKnow2	1 año	32 min. (real)
WorkKnow3	12 años	6 horas (estimado)

Tabla 10 - Ejecuciones de la generación de conocimiento en tres etapas

Como vemos, la estimación para la realización de los WorkKnow3 en tres etapas es elevada. Todo ello es debido a que se están tratando los ficheros de forma individual y la obtención de las listas de ficheros por parte del proceso que pertenecen al periodo de tiempo elegido es un proceso lento al solicitar los datos al *NameNode*.

5.2.4 Análisis espacial de los ficheros generados

En secciones anteriores se ha propuesto la implementación de una estructura de ficheros y directorios con un contenido bien definido y con una estructura adecuada para el tratamiento de las capturas de *streaming* que se realizan. En este punto se puede realizar una evaluación de sus requerimientos de espacio.

El *NameSpace* contiene información de ficheros, directorios y bloques. Los ficheros están divididos en bloques de gran tamaño. Para proporcionar disponibilidad, HDFS utiliza la replicación de los bloques, por defecto, en dos *DataNodes*. El *NameNode* mantiene el *NameSpace* totalmente en RAM. Esta arquitectura tiene un factor de limitación muy evidente: el tamaño de la memoria, esto es, el número de objetos en el *NameSpace* (ficheros, directorios y bloques) que se deben mantener.

Es el momento de verificar si nuestra estructura de datos es adecuada teniendo en cuenta este factor limitante. En [24] podemos encontrar los tamaños del mantenimiento en memoria de cada unos de los objetos que forman el sistema de ficheros HDFS. Por lo tanto en base a una valoración de los objetos que se han generados en las pruebas (ficheros, directorios y bloques) y una duración de diez años, podemos obtener la "Tabla 11 - Necesidades de memoria en *NameNode* (10 años)".

	# objetos almacenados	Bytes del objeto[24]	GB en memoria
Ficheros	70.080	250	0,16
Directorios	44.177	290	0,11
Bloques	4.064.640	368	13,93
Total	525.600	908	14,21

Tabla 11 - Necesidades de memoria en *NameNode* (10 años)

La estimación que se ha realizado es para el caso de bloques de 32 MB que, aunque no es el caso más probable, si es el más desfavorable. Para 10 años de datos almacenados, sería necesario mantener un sistema de ficheros que contendría 14,21 GB lo que representa un espacio elevado para almacenar los datos. Tengamos en cuenta que la mayor cantidad de almacenamiento es debida al número total de bloques que tenemos. La utilización de un sistema de ficheros con 128MB supondría utilizar una cuarta parte: 3,55GB. Esto es debido a que el 98% del espacio utilizado se debe a la utilización de gran cantidad de bloques. Debemos tener también en cuenta que el incremento del factor de replicación podría suponer un aumento considerable del espacio en memoria necesario.

La valoración del espacio total de almacenamiento requerido para la realización del proyecto se debe realizar teniendo en cuenta que existen dos replicas de todos los bloques que se graben. Por ello, el tamaño de los dos ficheros que se tienen en cuenta, el fichero de captura (*sequence file*) y el de resultado, con un tamaño de 467 MB de tamaño total hay que multiplicarlo por las 2 replicas, por los 365 días del año, por 24 horas y 4 ficheros por hora. El resultado es que es necesario 31 TB de almacenamiento para cada año capturado. Esta cifra está muy por encima de las capacidades del cluster de pruebas disponible.

Es aquí donde podemos confirmar que la utilización de los bloques virtuales es una mejor elección para el procesamiento de los ficheros que la utilización de la reducción física del tamaño de bloque. Se aúnan dos ventajas, la optimización del almacenamiento (ver párrafos anteriores) y una mayor adaptabilidad a las condiciones variables del trabajo (ver en la sección 5.1.). El estudio del reparto de los bloques entre los nodos (también en la sección 5.1.) no ha aportado información de que método es mejor para el tratamiento.

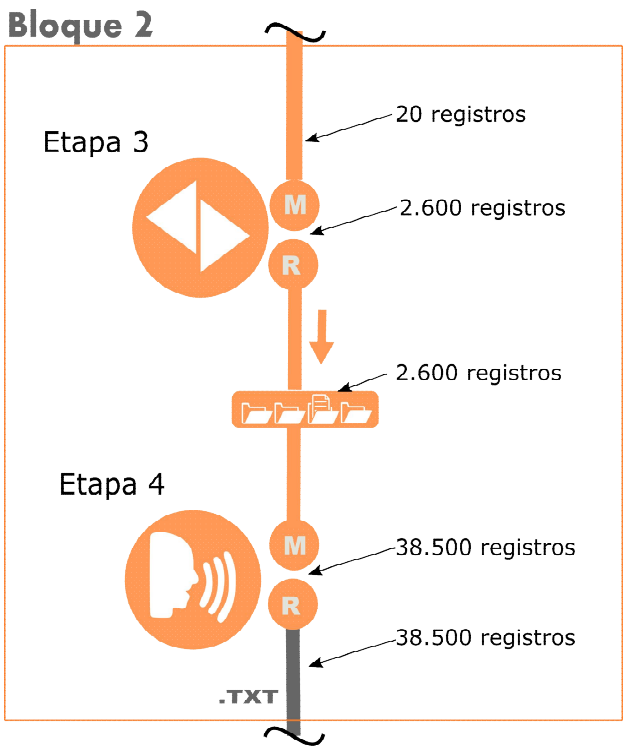

Figura 31 - Transferencia de registros en Bloque 2

Cada una de las parejas clave/valor que se almacenan en un *sequence file* constituyen un único registro que se lee en su totalidad cuando se accede a este. Si deseamos analizar los registros que se están transfiriendo entre los diferentes procesos MapReduce o incluso conocer las transferencias que existen entre la etapa 3 y la etapa 4 se pueden ver en la "Figura 31 - Transferencia de registros en Bloque 2". No importa el tamaño que tengan los bloques, en este caso el total de registros coincide con independencia del tamaño del bloque. Como se puede observar, el inicio corresponde con una captura de 20 registros (10 correspondientes a la captura de TV y 10 a emisiones de radio).

Si se han generado 38.500 registros en la ejecución de un fragmento de 15 min. el total de registros para un año puede estar próximo a los 1.000 millones de palabras, lo que constituye un problema de gran magnitud.

5.3 Análisis del conocimiento generado.

Es difícil poder hacer una valoración de los resultados obtenidos en la fase de generación del conocimiento ya que los datos utilizados no corresponden con capturas reales. Esto es, los datos que se han obtenido corresponden con el resultado del procesamiento de un juego de pruebas creado con los ficheros utilizados para entrenar los modelos de reconocimiento.

La analítica de datos de cargas de trabajo mixtas obliga a la utilización de un sistema integral de administración de flujos de trabajo. Todos los procesos se han podido tratar con un workflow compuesto por una fase inicial de selección del periodo de tiempo que deseamos tratar, una fase intermedia de procesamiento MapReduce que permite un paralelismo en el tratamiento y una última fase de presentación de los datos de forma gráfica. El tercer workflow, workflow3 - generación del mapa de calor, carece de la primera etapa de selección de ficheros. Esto se debe a la desastrosa estimación del tiempo de ejecución que se ha realizado previamente ("Tabla 10 - Ejecuciones de la generación de conocimiento"), por ello, la selección de los ficheros se hace en el proceso de Map con la utilización de un fichero con todas las referencias a los reconocimientos disponibles.

Los formatos de almacenamiento de información que se han diseñado en la fase de reconocimiento (bloque de tratamiento 2) no son adecuados para el tratamiento de grandes volúmenes de información como son necesarios en este caso (bloque de tratamiento 3). Aunque el rediseño del último tratamiento, workflow3, se ha realizado para utilizar lo mínimo posible los servicios que mantienen la estructura del sistema de ficheros, la paralelización ha sido pobre y la utilización de numerosos ficheros no permite un desempeño adecuado.

- **WorkKnow1. La generación de la nube de palabras**

  Para la valoración de la generación de la nube de palabras se ha simulado la utilización de un juego de pruebas de dos horas de duración. Esto corresponde a 8 ficheros de texto reconocidos previamente entre las capturas vocales el juego de entrenamiento. Esta primera aproximación es válida para valorar el correcto funcionamiento del sistema. Se han generado las nubes de palabras de una forma correcta que permitiría su utilización en un sistema en producción para conocer lo que se ha mencionado más en las últimas dos horas en los medios de comunicación. En nuestro caso, en la nube de palabras ("Figura 20 - Resultado ejecución WorkKnow1") que se muestra como resultado, se puede observar que existe una palabra muy utilizada (té) y otras que le siguen (pilastra, azul, ojeda, verde y pared).

- **WorkKnow2. Generación de gráfico temporal.**

  La capacidad de detectar la utilización de ciertas palabras durante la emisión puede dar lugar a obtener series temporales muy útiles para conocer la importancia de ciertos contenidos. Poder comparar las apariciones de una palabra o un grupo de palabras con las existentes en el año anterior parece un desarrollo útil. En el ejemplo presentado ("Figura 21 - Resultado ejecución WorkKnow2") se puede observar la frecuencia de aparición de dos palabras en un juego de pruebas. Se diferencia por la aparición en cuatro medios y por la totalización en función del tiempo.

- **WorkKnow3. Generación de mapa de calor.**

  Con esta tercera opción se ha generado un mapa de calor ("Figura 22 - Resultado ejecución WorkKnow3") con información temporal muy amplia, de varios años. La duración en el tiempo da una perspectiva bastante extensa de las apariciones de las palabras de muestra y de la importancia que ha tenido ese grupo de palabras durante años sucesivos. Es una manera de verificar la existencia de series temporales en las apariciones de palabras.

La conclusión a la que podemos llevar es que para diseñar procesos que suponen el tratamiento de periodos de tiempo pequeños (unas horas) el formato de almacenamiento descrito es adecuado. En el caso de querer realizar procesos que incluyan una duración temporal más amplia (varios años) se hace necesaria la utilización de un almacenamiento de datos más compacto en el que se reúna la mayor cantidad de información posible en un menor número de ficheros. No se descarta la realización de un proceso de fusión de varios ficheros de reconocimiento en un único fichero que contenga la información de reconocimiento de todo un mes, trimestre o año.

6 Conclusiones y Trabajos Futuros

En este proyecto se han analizado las vías que son utilizables, y aquellas que son necesarias, para poner en marcha un proyecto de reconocimiento vocal en un sistema batch con vistas a tener en cuenta no solo los documentos escritos sino los documentos sonoros con el objetivo de conocer el interés sobre algún tema. Se ha analizado la posibilidad y viabilidad de utilizar una infraestructura de bajo coste como es Hadoop permitiendo la democratización de estos métodos en numerosos ámbitos para el reconocimiento de grandes volúmenes de datos no textuales.

Se ha logrado el tratamiento de ficheros no textuales que generalmente son poco usados en la bibliografía y documentación de la plataforma Hadoop definiendose un proceso semiautomático para este tratamiento con la presentación de resultados comprensibles por los usuarios finales.

Conocer los límites de la mejora que se obtiene con la utilización de un cluster Hadoop en función del tamaño de bloque que se le asigna a cada uno de los procesos, bien por tamaño de bloque físico o virtual, permitirá la adaptación de la dimensión del cluster a las necesidades de computo. Pero también se ha podido entender la necesidad de almacenar los datos en formatos de ficheros alternativos, óptimos para su tratamiento, diferentes del tipo texto que son utilizados habitualmente en los tratamientos con Hadoop.

Se han estudiado las vías para la integración de un gestor de workflow que realice la ejecución en función de la existencia de los ficheros de captura y se han generado algunos gráficos significativos con información útil que son capaces de expresar conocimiento en base a datos obtenidos de los archivos multimedia.

Se ha analizado el desempeño del sistema y se han obtenido los límites de mejora, valorándose el espacio necesario tanto en los elementos propios de la infraestructura como en aquellos que deben acomodar la información de captura o resultado. Se han conseguido tiempos de respuesta del sistema que permiten el tratamiento de fuentes de datos de gran volumen.

La utilización de un sistema distribuido batch como es Hadoop limita considerablemente la capacidad de respuesta de las aplicaciones a las variaciones que puedan existir. La generación del conocimiento hace necesaria la utilización de estructuras de datos más adaptadas siendo estas diferentes a las necesarias en el proceso de captura y reconocimiento.

Como primera propuesta de trabajo futuro y con vistas al verificar el comportamiento integro de la totalidad del sistema, se propone la obtención de un modelo vocal entrenado válido acorde con los ficheros de audio tratados que permita el reconocimiento con ciertas garantidas de las capturas que se realicen.

Otra posible vía de ampliación del presente trabajo sería la utilización de otras fuentes de datos. El uso de plataformas de alojamiento de vídeos, como YouTube, o la utilización del tratamiento de la VoIP podrían ser dos vías de mejora de la fuentes. Todo esto llevaría a una posible internacionalización de la aplicación que podrían utilizar diferentes fuentes de datos tanto nacionales como extranjeras en diferentes idiomas sin dejar de lado las regionales. Esta mejora supondría la utilización de diferentes tipos de modelos de reconocimiento para diferentes idiomas y una ampliación a la traducción automática de los textos.

Existen algunos análisis con los que se podría completar el estudio de las capturas que se realizan, por ejemplo el análisis de sentimientos, descubrimiento de la localización o personalización del hablante o del referenciado, pero en cualquier caso son comunes tanto si la fuente del datos es de tipo texto o bien en vocal.

Otra mejorar y posible trabajo es la comparación de esta sistema batch con un sistema en tiempo real basado en alguna plataforma como Storm o Kafka que puede mejorar la experiencia de trabajo de un usuario proporcionando información en tiempo real.

7 Bibliografía

1. AudioSeg Site. Web: *https://gforge.inria.fr/projects/audioseg*
2. ALIZE Site. Web: *http://mistral.univ-avignon.fr/index_en.html*
3. Aminzadeh, A. R. & Shen, W. (2009). "*Advocate: A Distributed Architecture for Speech-to-Speech Translation*". Lincoln Laboratory Journal, 18(1). 2009.
4. Azkaban Site. Web: http://azkaban.github.io/
5. BigR Site. Web: https://developer.ibm.com/hadoop/docs/biginsights-value-add/big-r/
6. BlueMix Site. Web: *https://console.ng.bluemix.net/*
7. Bonastre, J.; F., Wils, F.; Meignier, S.:"*ALIZE, a free toolkit for speaker recognition*". In ICASSP (1) (pp. 737-740). 2005. Web: *http://mistral.univ-avignon.fr/doc/publi/05_Interspeech_Bonastre.pdf*
8. Cassidy, S.: "*COMP449: Speech Recognition*".Department of Computing, Macquarie University, Australia, 2002. Web: *http://web.science.mq.edu.au/~cassidy/comp449/html/ch14.html*
9. Chong, J.; Friedland, G.; Janin, A., Morgan, N. & Oei, C. "*Opportunities and challenges of parallelizing speech recognition*". InProceedings of the 2nd USENIX conference on Hot topics in parallelism, Berkeley, CA, USA (p. 85).June 2010.
10. "*Cisco visual Networking Index: Global Mobile data Traffic Forecast Update, 2013-2018*", Cisco 2014. Web: *http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html*
11. Cloudera Site. Web: http://www.cloudera.com/
12. Dean, J. & Ghemawat, S.: "*MapReduce: Simplified Data Processing on Large Clusters*". Comunications of the ACM, 51(1), 107-113. 2004.
13. Ekanayake, J.; Li, H. ; Zhang, B.; Gunarathne, T. S.; Bae, H.; Qiu, J. & Fox, G.: "*Twister: a runtime for iterative mapreduce*" in HPDC, 2010, pp. 810–818.
14. Ezzatti, P. M.; Fernández, E.: "*Trabajos preliminares sobre radiosidad y paralelismo*". Reportes Técnicos 09-05. Universidad de la República Montevideo, Uruguay. 2009.
15. Eclipse Site. Web: *https://www.eclipse.org/*
16. Fadika, Z.; Dede, E.; Govindaraju, M. & Ramakrishnan, L.: "*Benchmarking mapreduce implementations for application usage scenarios*". InGrid Computing (GRID), 2011 12th IEEE/ACM International Conference on (pp. 90-97). IEEE. Web: *http://escholarship.org/uc/item/35b550ph#page-2*
17. Fadika, Z.;Govindaraju, M.: "*Lemo-mr: Low overhead and elastic mapreduce implementation optimized for memory and cpu-intensive applications*". Cloud Computing Technology and Science, IEEE International Conference on, vol. 0, pp. 1–8, 2010.
18. Ffmpeg Site. Web: *https://www.ffmpeg.org/*
19. Friedland, G.; Chong, J. & Janin, A.:"*A parallel meeting diarist*" In Proceedings of the 2010 international workshop on Searching spontaneous conversational speech (pp. 57-60). ACM. Oct 2010.
20. Galliano, S; Gravier, G.;Chaubard, L.: "*The ESTER 2 Evaluation Campaign for the Rich Transcription of French Radio Broadcasts*". Interspeech 2009. Brighton. Web : *http://www.irisa.fr/metiss/ggravier/biblio/09/galliano-interspeech-09.pdf*
21. Ghemawat, S.; Gobioff, H.; Leung, S.:"The Google File System". 19th ACM Symposium on Operating Systems Principles. Oct 2003
22. Gravier, G.; Betser, M.; Mathieu B.: "*Audioseg. Audio Segmentation Toolkit rel. 2.1*". IRISA – Institut de Recherche en Informatique et Systèmes Alèatoires. 2010.
    Web: *https://gforge.inria.fr/frs/download.php/25187/audioseg-1.2.pdf*
23. Gilgoff, D.; Lee,J.J.: "*National Geographic News. Social Media Shapes Boston Bombings Response*". National Geographic News. 2003. Web: *http://news.nationalgeographic.com/news/2013/13/130415-boston-marathon-bombings-terrorism-social-media-twitter-facebook/*
24. Hadoop Common Board. Hadoop-1687 Name-node memory size estimates and optimazation proposal. Web: *https://issues.apache.org/jira/browse/HADOOP-1687*
25. Hadoop On The Road Blog. Web: *http://hadoopontheroad.blogspot.com.es/search/label/Java*
26. Hadoop Web Page: Web: *http://hadoop.apache.org/*
27. HDFS Architecture Site: Web: *http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html*
28. Holmes, A.: "*Hadoop in Practice*". Manning. 2012.
29. Hooked on Hadoop Blog. Web*: http://hadooped.blogspot.com.es/*
30. Hortonworks Site. Web: *http://hortonworks.com/*
31. IBM InfoSphere BigInsights. Web: *http://www.ibm.com/developerworks/data/library/techarticle/dm-1110biginsightsintro/*

32. Ihaka, R.: "*R: Past and future history*". Computing Science and Statistics, 392-396. 1998 Web: *https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf*
33. Ihaka, R. & Gentleman, R. "*R: a language for data analysis and graphics*". Journal of computational and graphical statistics, 5(3), 299-314. 1996. Web: *https://www.stat.auckland.ac.nz/~ihaka/downloads/R-paper.pdf , http://www-users.york.ac.uk/~pml1/r/ig.pdf*
34. JULIUS Site. Web: *http://julius.sourceforge.jp/en_index.php?q=index-en.html#whats_new*
35. jzawodn: "*Yahoo! Launches Word's Largest Hadoop Production Application*" Yahoo Developer Network. 2008. Web. *https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html*
36. Khamadi Were, Daudi: "*How Kenya turned to social media aftger mall attack. CNN international edition.*" 25 sept 2013 CNN News. Web: *http://edition.cnn.com/2013/09/25/opinion/kenya-social-media-attack/*
37. Kim, M.; Han,S.:Cui,Y. & Jeong, C.: "*A Hadoop-based Multimedia Transcoding System for Procesing Social Media in the PaaS Platform of SMCCSE*". KSII Transactios on Internet and Information Systems. 2012. Web: *http://dcslab.konkuk.ac.kr/papers/TIIS_Vol6No11P5Nov2012.pdf*
38. Kotti, M.; Moschou, V. & Kotropoulos, C.: "*Speaker segmentation and clustering*". Signal processing, 88(5), 1091-1124. 2008
39. Laboratoire d'Informatique de l'Université du Maine website. Web: *http://www-lium.univ-lemans.fr*
40. LIUM Diarization Web Site: *http://www-lium.univ-lemans.fr/diarization/doku.php/*
41. "*Marco General de los medios en España 2014*" AIMC – Asociacion para la Investigación de Medios de Comunicación. 2014 Web: *http://www.aimc.es/spip.php?action=acceder_document&arg=2477&cle=8ef27792fa0ef790b25becb87db929013d9949e9&file=pdf%2Fmarco14.pdf*
42. Luigi Site. Web: *https://github.com/spotify/luigi*
43. MapR Site. Web: *https://www.mapr.com/*
44. Meignier,S.; Merlin, T.: "*LIUM SpkDiarization: An Open Source Toolkit For Diarization*". Proc. CMU SPUD Workshop, March 2010, Dallas (Texas, USA).
45. Mplayer web page. Web: *http://www.mplayerhq.hu/design7/dload.html*
46. Muñoz, L; Urueña, A. & alt "*Los Contenidos Digitales en España. Informe Anual 2011*". ONTSI - Observatorio Nacional de las Telecomunicaciones y de la SI. 2012. Web: *http://www.ontsi.red.es/ontsi/sites/default/files/informe_contenidos_digitales_edicion2012.pdf*
47. Niyizamwiyitira, C. & Lundberg, L. "*Performance evaluation and prediction of open source speech engine on multicore processors*". InProceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems (pp. 345-352). ACM. Oct.2013. Web: *http://www.bth.se/fou/Forskinfo.nsf/0/1802b8b413d0322cc1257cba002deba0/$file/Performance%20evaluation%20and%20prediction.pdf*
48. Oozie Site. Web: *http://oozie.apache.org/*
49. Paraskevas, A. & Altinay, L. "*Signal detection as the first line of defence in tourism crisis management*". Tourism Management, 34, 158-171. 2013 Web: *http://www.academia.edu/2765341/Signal_detection_as_the_first_line_of_defence_in_tourism_crisis_management*
50. Rouvier, M., Dupuy, G., Gay, P.; Khoury,E,; Merlin,T. & Meignier, S.: "*An Open-source State-of-the-art Toolbox for Broadcast News Diarization*" Interspeech, Lyon (France), 25-29 Aug. 2013 Web: *http://infoscience.epfl.ch/record/192762/files/Rouvier_INTERSPEECH_2013.pdf*
51. Saracco, C: "*Understanding InfoSphere BigInsights: An introduction for software architects and technical leaders*". IBM developerWorks, Oct 2011. Web: *http://www.ibm.com/developerworks/data/library/techarticle/dm-1110biginsightsintro/dm-1110biginsightsintro-pdf.pdf*
52. SHoUT website. Web: *http://shouttoolkit.sourceforge.net/use_case_diarization.html*
53. Stuart Sierra Blog. Sequence file code generator TarToSeqFile. Web: *http://stuartsierra.com/2008/04/24/a-million-little-files*
54. Tan, Y. S. & Lee, B. S.: "*MapReduce Framework: Some Challenges*". In Annual International Conference on CCV (pp. 278-284). 201.Web: *http://www.globalstf.org/docs/proceedings/ccv/188.pdf*
55. VoxForge Site. Web: *http://www.voxforge.org/*
56. Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E. & Woelfel, J.:"*Sphinx-4: A flexible open source framework for speech recognition*" Technical Report Sun Microsystems, Inc. 2004 Web: *http://www.researchgate.net/publication/228770826_Sphinx-4_A_flexible_open_source_framework_for_speech_recognition/file/79e4150c20aeb37c52.pdf*
57. White, T.: "*Hadoop: The definitive guide*" Ed. O'Reilly Media, Inc.", 2012.
58. You, K.; Chong, J.; Yi, Y.; Gonina, E.; Hughes, C. J.; Chen, Y. K. & Keutzer, K.: "*Parallel scalability in speech recognition*". Signal Processing Magazine, IEEE, 26(6), 124-135. 2009.

8 Anexos

8.1 Anexo I - Ejemplo de implementación para el bloque 1

```
#!/bin/bash

captura_a3(){
rtmpdump -m 200 -r "rtmp://antena3fms35livefs.fplive.net:1935/antena3fms35live-live" -y "stream-
antena3_1" -W "http://www.antena3.com/static/swf/A3Player.swf?nocache=200" -p
"http://www.antena3.com/directo/" -q -v > $DIR/A3"-"$TIME &
}

captura_24h(){
#no va
rtmpdump -m 200 -r "rtmp://rtvefs.fplive.net:1935/rtve-live-live?ovpfv=2.1.2" -y
"RTVE_24H_LV3_WEB_NOG" -W "http://www.irtve.es/swf/4.2.15/RTVEPlayerVideo.swf" -p
"http://www.rtve.es"  -C S:OK -q -v > $DIR/rtve24h"-"$TIME &

}
captura_tv3(){
# va
rtmpdump -v -r "rtmp://tv-nogeo-flashlivefs.fplive.net/tv-nogeo-flashlive-live/stream_TV3CAT_FLV"
> $DIR/TV3"-"$TIME &
}
captura_ribera(){
rtmpdump -m 200 -r "rtmp://flash3.todostreaming.es/ribera" -y "livestream" -W
"http://www.todostreaming.es/player_new.swf" -p "http://www.riberatelevisio.com" > $DIR/RiberaTV"-
"$TIME  &
}
captura_lasexta(){
 rtmpdump -m 200 -r "rtmp://antena3fms35livefs.fplive.net:1935/antena3fms35live-live/stream-
lasexta_1" -W "http://www.antena3.com/static/swf/A3Player.swf" -p "http://www.lasexta.com/directo"
-q -v > $DIR/la6"-"$TIME &
}

captura_cope(){
mplayer -noframedrop -dumpstream http://94.75.206.136/tunein.php/radioaranda/playlist.pls -
dumpfile $DIR/cope"-"$TIME".mp3" > /dev/null &
}

captura_rne1(){
mplayer -noframedrop -dumpstream http://radio1.rtve.stream.flumotion.com/rtve/radio1.mp3.m3u -
dumpfile $DIR/rne1"-"$TIME".mp3" > /dev/null &
}
captura_rne5(){
mplayer -noframedrop -dumpstream http://radio5.rtve.stream.flumotion.com/rtve/radio5.mp3.m3u -
dumpfile $DIR/rne5"-$TIME.mp3" > /dev/null &
}
captura_ser_v(){
mplayer -noframedrop -dumpstream http://4123.live.streamtheworld.com:80/SER_VALENCIA_SC -dumpfile
$DIR/serVAL"-$TIME.mp3" > /dev/null &
}
captura_radio_exterior(){
#va
mplayer -noframedrop -dumpstream
http://radioexterior.rtve.stream.flumotion.com/rtve/radioexterior.mp3.m3u -dumpfile
$DIR/radioexterior-"$TIME.mp3" > /dev/null &
}

DIR=./test
CONV=conv

DAY=$(date '+%d')
MONTH=$(date '+%m')
YEAR=$(date '+%Y')
HOUR=$(date '+%H')
M=$(date '+%M')

ls -la $DIR
echo $DIR ; rm  $DIR/*; mkdir $DIR
```

```
echo $CONV ; rm  $CONV/*;mkdir $CONV

if   [[ "$M" < "15" ]] ; then Q=00
elif [[ "$M" < "30" ]] ; then Q=15
elif [[ "$M" < "45" ]] ; then Q=30
else Q=45
fi


TIME=$YEAR"-"$MONTH"-"$DAY"-"$HOUR"-"$Q
echo "Quarter: $Q Day:$DAY Month:$MONTH Year:$YEAR Hour:$HOUR  FileNames: $TIME"

echo "============================================================"
echo "===        CAPTURA  ETAPA1          ====================="
echo "============================================================"

{
captura_a3 ; captura_levanteTV; captura_tv3;captura_ribera;captura_lasexta
captura_cope;captura_rne1;captura_rne5;captura_ser_v;captura_radio_exterior
} &>./temp_captura


echo "tiempo captura $1 sec"

if [ "$1" == "" ]; then
   tiempo=10
else
   tiempo=$1
fi

sleep $tiempo
killall rtmpdump ;killall mplayer

echo "============================================================"
echo "===        CONVERSION    ETAPA2     ====================="
echo "============================================================"

echo
cd $DIR
echo $CONV
for i in $( ls ); do
    base=${i%%.*}
    echo item: $base
    ffmpeg -i $i -acodec pcm_s16le -ac 1 -ar 16000 ../$CONV/$base.wav  &>./temp_conv
done
cd ..

echo "============================================================"
echo "===        Generacion sequence file ====================="
echo "============================================================"

SEQ_NAME="base-"$TIME".sf"

tar -cvf output.tar ./$CONV
java -jar tar-to-seq.jar output.tar $SEQ_NAME

rm output.tar
rm .base*crc
```

8.2 Anexo II - Extracción de sequence files

```
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.SequenceFile;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.util.ReflectionUtils;

public class read_sequencefile {
public static void main(String[] args) {
       try{
              Configuration conf = new Configuration();
              FileSystem fs = FileSystem.getLocal(conf);
              Path p = new Path("../NewsASR/output/stage1/part-r-00000");
              SequenceFile.Reader reader = new SequenceFile.Reader(fs,p,conf);
              Writable key = (Writable) ReflectionUtils.newInstance(reader.getKeyClass(), conf);
              BytesWritable value= ReflectionUtils.newInstance(reader.getValueClass().asSubclass(
                     BytesWritable.class),conf);
              long position = reader.getPosition();
              while (reader.next(key, value)) {
                     String syncSeen = reader.syncSeen() ? "*" : "";
                     System.out.println("Mostrar: "+ key
                                          + " posicion :" + position );
                     position = reader.getPosition();
                     InputStream is = new DataInputStream(
new ByteArrayInputStream(value.getBytes()));
                     AudioInputStream in =
                             AudioSystem.getAudioInputStream(is);
                     File f=new File(key.toString());
                     System.out.println(f.getName());
                     OutputStream out = new FileOutputStream("./"+f.getName().toString());
                     AudioSystem.write( in, AudioFileFormat.Type.WAVE,   out);
                     out.close();
                     in.close();
                     is.close();
                     in.reset();
                     is.reset();
       }
       }catch(Exception e){
              e.printStackTrace();
       }
} }
```

8.3 Anexo III - MapReduce Etapa 3

MapReduce

```
package com.NewsASR;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Iterator;
import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import org.apache.commons.io.FilenameUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableUtils;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;


public class NewsASRstage1 {

public static class NewsASRmapper extends Mapper<Text, BytesWritable, Text, BytesWritable> {
       public void map(Text key, BytesWritable value, Context context)
              throws IOException, InterruptedException {

       System.out.println("================================================================");
       System.out.println("=====================  MAPPER stage1   =======================");
       System.out.println("============ Classname: " + this.getClass().getName() +"=======");
       System.out.println("================================================================");
       Configuration conf = context.getConfiguration();
       String output_dir = conf.get("output.dir");
       System.out.println("Output Dir: " + output_dir);
       boolean verbose = Boolean.parseBoolean(conf.get("verbose.mode"));
       System.out.println ("Verbose:" + verbose);
       try {
              NewsASRTools conversor = new NewsASRTools();
              InputStream is = new DataInputStream(new ByteArrayInputStream(value.getBytes()));
              AudioInputStream in = AudioSystem.getAudioInputStream(is);
              System.out.println("READING: Key->(" + key + ")\t  value->("
                                   + in.getFormat().toString() + "\t size:"
                                   + value.getLength() + ")");
              String basename = FilenameUtils.getBaseName(key.toString());
              String extension = FilenameUtils.getExtension(key.toString());
              conversor.ShowInfo(in);
              /*******************************************************
               *
               *
               * Convert format
               *
               *
               *******************************************************/
              //conversion, it is not necessary
              //AudioInputStream in_conv = conversor.convert(in);
              AudioInputStream in_conv = in;
              /*******************************************************
```

```
 *
 *

 * Diarization
 *
 *
 ******************************************************/
conversor.runcommand("pwd", true);
// Create shell for diarization
conversor.genera_shell("./diarizacion.sh");
conversor.runcommand("chmod 777 ./diarizacion.sh", false);
// Save file for diarization
OutputStream out = new FileOutputStream("./test_mapper.wav");
AudioSystem.write(in_conv, AudioFileFormat.Type.WAVE,      out);
out.close();
in_conv.reset();
//Delete diarization subdirectory
System.out.print("Deleting temporary directory.....");
String temp_dir="./test_mapper";
File folder = new File(temp_dir);
if (folder.exists()) {
       folder.delete();
}
System.out.println("Deleted");
// Diarization
System.out.println("Start Diarization");
String comando="./diarizacion.sh ./test_mapper.wav";
conversor.runcommand(comando, false);
System.out.println("End Diarization");
String splitBy = " ";
//Reading diarization results
BufferedReader br = new BufferedReader(
                     new FileReader(temp_dir + "/test_mapper.s.seg"));
String line = null;
int counter=0;
while ((line = br.readLine()) != null) {
       System.out.println (">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>");
       counter++;
       String[] b = line.split(splitBy);
       if (!b[0].toString().isEmpty()) {
       if (b[0].toString().trim().charAt(0) != ';') {
              System.out.println("Wav slice generation: Start "
                           + Double.parseDouble(b[2].toString())/ 100.0
                           + "sec. Length: "
                           + Double.parseDouble(b[3].toString()) / 100.0
                           + "sec.");
              //New key string with start and length time
              Text new_key=new Text( basename
              +"_"+String.format("%06d",  Long.parseLong(b[2].toString()) )
              +"_"+String.format("%06d",  Long.parseLong(b[3].toString()) ) );
              in_conv.reset();
              // Cut file
              conversor.copyAudio(   "./test_mapper.wav",
                                     "./post_cut.wav",
                                     Double.parseDouble(b[2].toString())*10,
                                     Double.parseDouble(b[3].toString())*10);


              //Save temporary file in verbose mode
              if (verbose){
                     String coman="cp ./post_cut.wav ./test_"+new_key+".wav";
                     System.out.println ("comando:" + coman);
                     conversor.runcommand(coman, true);
              }
              InputStream audioInputStream = new DataInputStream(
                                   new FileInputStream("./post_cut.wav"));
              BufferedInputStream bufferedInputStream
                            = new BufferedInputStream(audioInputStream);
              AudioInputStream in_conv_cutted2 =
                     AudioSystem.getAudioInputStream(bufferedInputStream);
              conversor.ShowInfo(in_conv_cutted2);
              BytesWritable bw2 = conversor.to_bytesWritable(in_conv_cutted2,conf);
              System.out.println("Tamaño writeSizable:"
                                   + WritableUtils.toByteArray(bw2).length);
              bw2.setCapacity(bw2.getLength());
              // Send to Reduce
              context.write(new_key, bw2);
```

```
                        }}
                }
                //close diarization results file
                br.close();
                // If diarization return 0 lines
                if (counter==0) {
                        // Put all file length
                        System.out.println("Full slice");
                        Text new_key=new Text(basename+"_"+"000ALL" + "_"  + "000ALL" );
                        BytesWritable bw2 = conversor.to_bytesWritable(in_conv,conf);
                        context.write(new_key, bw2);
                }

        } catch (Exception e) {
                e.printStackTrace();
        }
}}



public static class NewsASRreducer extends Reducer<Text, BytesWritable, Text, BytesWritable> {
public void reduce(Text key, Iterable<BytesWritable> values, Context context)
                        throws IOException, InterruptedException {
        System.out.println("============================   REDUCER    ====================");
        System.out.println("============= Classname: " + this.getClass().getName() + "=====");
        Configuration conf= context.getConfiguration();
        boolean verbose =  Boolean.parseBoolean(conf.get("verbose.mode"));
        System.out.println ("Verbose:" + verbose);
        String output_dir=conf.get("output.dir");
        if (verbose){
                System.out.println("Output Dir: " + output_dir);
        }
        try{
                // foreach value in list
                Iterator<BytesWritable> itr = values.iterator();
                while (itr.hasNext()) {
                        //get binary part
                        BytesWritable value = new BytesWritable(itr.next().getBytes());
                        value.setCapacity(value.getLength());
                        InputStream is = new DataInputStream(
                                        new ByteArrayInputStream(value.getBytes()));
                        AudioInputStream in = AudioSystem.getAudioInputStream(is);
                        if (verbose){
                                //Print key and information
                                System.out.println(key + "  ByteWritable Size :"
                                        + WritableUtils.toByteArray(value).length
                                        + " Seconds:"
                                        + (in.getFrameLength() + 0.0D)
                                                / in.getFormat().getFrameRate());
                        }
                        //write in sequence file
                        context.write(key, value);
                }
        }catch(Exception e){
                e.printStackTrace();
        }
}
} // End NewsASRreducer



public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                String[] programArgs = new GenericOptionsParser(conf, args)
                                .getRemainingArgs();
                System.out.println("Total argumentos:" + programArgs.length);
                for (int n = 0; n < programArgs.length; n++) {
                        System.out.println(programArgs[n]);
                }
                if (programArgs.length != 2) {
                        System.exit(-1);
                }
                String input_file = programArgs[0];
                String output_dir = programArgs[1];
                System.out.println("Input Sequence File:" + input_file);
                System.out.println("Output Dir         :" + output_dir);
```

```
                conf.set("output.dir", output_dir);
                conf.set("input_file", input_file);
                conf.set("verbose.mode","true");
                Job job = new Job(conf);
                job.setJarByClass(NewsASRstage1.class);
                job.setMapperClass(NewsASRmapper.class);
                job.setReducerClass(NewsASRreducer.class);
                job.setInputFormatClass(SequenceFileInputFormat.class);
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(BytesWritable.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(BytesWritable.class);
                job.setOutputFormatClass(SequenceFileOutputFormat.class);
                job.setNumReduceTasks(1);
                FileInputFormat.addInputPath(job, new Path(input_file));
                FileOutputFormat.setOutputPath(job, new Path(output_dir));
                job.waitForCompletion(true);
        }}
```

Workflow.xml Etapa 3 y Etapa 4

```
<workflow-app name="NewsASR" xmlns="uri:oozie:workflow:0.2">
<start to="stage1"/>
        <action name='stage1'>
                <map-reduce>
                        <job-tracker>${jobTracker}</job-tracker>
                        <name-node>${nameNode}</name-node>
                        <prepare>
                                <delete path="${nameNode}/${output_dir}/stage1"/>
                        </prepare>
                        <configuration>
                                <property>
                                        <name>mapred.job.name</name>
                                        <value>NewsASR</value>
                                </property>
                                <property>
                                        <name>mapreduce.map.class</name>
                                        <value>com.NewsASR.NewsASRstage1$NewsASRmapper</value>
                                </property>
                                <property>
                                        <name>mapreduce.reduce.class</name>
                                        <value>com.NewsASR.NewsASRstage1$NewsASRreducer</value>
                                </property>
                                <property>
                                        <name>mapred.output.dir</name>
                                        <value>${output_dir}/stage1</value>
                                </property>
                                <property>
                                        <name>output.dir</name>
                                        <value>${output_dir}/stage1</value>
                                </property>
                                <property>
                                        <name>mapred.input.dir</name>
                                        <value>${sequence_file}</value>
                                </property>
                                <property>
                                        <name>mapreduce.inputformat.class</name>
                        <value>org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat</value>
                                </property>
                                <property>
                                        <name>mapreduce.mapinput.key.class</name>
                                        <value>org.apache.hadoop.io.Text</value>
                                </property>
                                <property>
                                        <name>mapreduce.mapinput.value.class</name>
                                        <value>org.apache.hadoop.io.BytesWritable</value>
                                </property>
                                <property>
                                        <name>mapred.mapoutput.key.class</name>
                                        <value>org.apache.hadoop.io.Text</value>
                                </property>
                                <property>
                                        <name>mapred.mapoutput.value.class</name>
                                        <value>org.apache.hadoop.io.BytesWritable</value>
                                </property>
```

```
                        <property>
                                <name>dfs.block.size</name>
                                <value>${blockSize}</value>
                        </property>
                        <property>
                                <name>mapreduce.job.output.key.class</name>
                                <value>org.apache.hadoop.io.Text</value>
                        </property>
                        <property>
                                <name>mapreduce.job.output.value.class</name>
                                <value>org.apache.hadoop.io.BytesWritable</value>
                        </property>
                        <property>
                                <name>mapreduce.job.outputformat.class</name>
            <value>org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat</value>
                        </property>
                        <property>
                                <name>mapred.reducer.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapred.mapper.new-api</name>
                                <value>true</value>
                        </property>
                </configuration>
<archive>/user/biadmin/newsASR/lib/LIUM_SpkDiarization-8.4.jar#LIUM_SpkDiarization.jar</archive>
            </map-reduce>
            <ok to="stage2"/>
            <error to="kill"/>
        </action>

        <action name='stage2'>
                <map-reduce>
                        <job-tracker>${jobTracker}</job-tracker>
                        <name-node>${nameNode}</name-node>
                        <prepare>
                                <delete path="${nameNode}/${output_dir}/stage2"/>
                        </prepare>
                        <configuration>
                                <!-- <property>
                                        <name>mapred.max.split.size</name>
                                        <value>33554432</value>
                                </property> -->
                                <property>
                                        <name>mapred.job.name</name>
                                        <value>NewsASR</value>
                                </property>
                                <property>
                                        <name>mapreduce.map.class</name>
                                        <value>com.NewsASR.NewsASRstage2$NewsASRmapper</value>
                                </property>
                                <property>
                                        <name>mapreduce.reduce.class</name>
                                        <value>com.NewsASR.NewsASRstage2$NewsASRreducer</value>
                                </property>
                                <property>
                                        <name>mapred.output.dir</name>
                                        <value>${output_dir}/stage2</value>
                                </property>
                                <property>
                                        <name>output.dir</name>
                                        <value>${output_dir}/stage2</value>
                                </property>
                                <property>
                                        <name>mapred.input.dir</name>
                                        <value>${output_dir}/stage1/part-r-00000</value>
                                </property>
                                <property>
                                        <name>mapreduce.inputformat.class</name>
                        <value>org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat</value>
                                </property>
                                <property>
                                        <name>mapreduce.mapinput.key.class</name>
                                        <value>org.apache.hadoop.io.Text</value>
                                </property>
                                <property>
                                        <name>mapreduce.mapinput.value.class</name>
```

```
                                <value>org.apache.hadoop.io.BytesWritable</value>
                        </property>
                        <property>
                                <name>mapred.mapoutput.key.class</name>
                                <value>org.apache.hadoop.io.Text</value>
                        </property>
                        <property>
                                <name>mapred.mapoutput.value.class</name>
                                <value>org.apache.hadoop.io.LongWritable</value>
                        </property>
                        <property>
                                <name>mapred.reducer.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapred.mapper.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapreduce.job.user.classpath.first</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapreduce.task.classpath.user.precedence</name>
                                <value>true</value>
                        </property>
                </configuration>
                <archive>/user/biadmin/newsASR/lib/asr.tar.gz#asr.tar.gz</archive>
        </map-reduce>
        <ok to="end"/>
        <error to="kill"/>
    </action>
    <kill name="kill">
        <message>error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

8.4 Anexo IV - MapReduce Etapa 4

MapReduce

```
package com.NewsASR;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import org.apache.commons.io.FilenameUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import com.google.common.io.*;

public class NewsASRstage2 {
public static class NewsASRmapper extends
       Mapper<Text, BytesWritable, Text, LongWritable> {
public void map(Text key, BytesWritable value, Context context)
              throws IOException, InterruptedException {
       System.out.println("============================================================================");
       System.out.println("======================   MAPPER stage 2   ==========================");
       System.out.println("========= Classname: " + this.getClass().getName() + "=============");
       System.out.println("============================================================================");
       Configuration conf = context.getConfiguration();
       String output_dir = conf.get("output.dir");
       System.out.println("Output Dir: " + output_dir);
       boolean verbose =  Boolean.parseBoolean(conf.get("verbose.mode"));
       System.out.println("Verbose:" + verbose);
       String temp_file="./" + key +"_o.wav";
       String result_file="./output.csv";
       try {
              NewsASRTools conversor = new NewsASRTools();
              InputStream is = new DataInputStream(new ByteArrayInputStream(value.getBytes()));
              AudioInputStream in = AudioSystem.getAudioInputStream(is);
              OutputStream out = new FileOutputStream(temp_file);
              AudioSystem.write(in, AudioFileFormat.Type.WAVE,out);
              out.close();
              System.out.println("READING: Key->(" + key + ")\t  value->("
                                          + in.getFormat().toString() + "\t size:"
                                          + value.getLength() + ")");
              String basename = FilenameUtils.getBaseName(key.toString());
              if (verbose){
                     System.out.println("basename:" + basename );
                     conversor.ShowInfo(in);
              }
              boolean local_exec =  Boolean.parseBoolean(conf.get("local.execution"));
              boolean show=false;
              /*********************************************************
               * ASR
               *********************************************************/
              ASR recognizer= new ASR(new ByteArrayInputStream(value.getBytes()),
                                  "esp", "sphinx4",verbose,local_exec);
              recognizer.run();
```

```
                String line2 = null;
                while ((line2 = recognizer.next()) != null) {
                        String new_key= key + "_" +recognizer.get_t_ini()+ "_"+
                                recognizer.get_t_fin() + "_" + recognizer.get_word() +"_";
                        if (verbose){
                                System.out.println("key -> " + new_key);
                        }
                        context.write(new Text(new_key), new LongWritable(1));
                }
                recognizer.close();
                if(false){
                if (verbose){
                        show=true;
                }
                if (!local_exec){
                        System.out.println("Running in server mode");
                        if (verbose){
                                conversor.runcommand("pwd",true);
                                conversor.runcommand("ls -ltr",true);
                        }
                        File f_result = new File(result_file);
                        if (f_result.exists()){
                                f_result.delete();
                        }
                        File f_run_origin = new File("./asr.tar.gz/run.sh");
                        File f_run_dst    = new File ("./run.sh");
                        if (!f_run_dst.exists()){
                                Files.copy(f_run_origin, f_run_dst);
                        }
                        File f_rec_origin = new File("./asr.tar.gz/reconoce.jar");
                        File f_rec_dst    = new File ("./reconoce.jar");
                        if (!f_rec_dst.exists()){
                                Files.copy(f_rec_origin, f_rec_dst);
                        }
                        File f_vox_origin = new File("./asr.tar.gz/voxforge");
                        File f_vox_dst    = new File ("./voxforge.tar.gz");
                        if (!f_vox_dst.exists()){
                                Files.copy(f_vox_origin, f_vox_dst);
                        }
                        conversor.runcommand("tar -xzvf voxforge.tar.gz",show);
                        conversor.runcommand("chmod 777 ./run.sh", show);
                }
                // Run ASR
                conversor.runcommand("./run.sh "+ temp_file,show);
                System.out.print("Deleting temporary wav file.....");
                File file_aux_temp = new File(temp_file);
                if (file_aux_temp.exists()) {
                        file_aux_temp.delete();
                        System.out.println("Deleted");
                }else{
                        System.out.println("NotExist");
                }
                // Reading asr result file
                String splitBy = ";";
                BufferedReader br = new BufferedReader(new FileReader(result_file));
                String line = null;
                while ((line = br.readLine()) != null) {
                        String[] b = line.split(splitBy);
                        String t_ini = b[0].toString();
                        String t_fin = b[1].toString();
                        String word  = b[2].toString().trim();
                        //new key
                        String new_key= key + "_" +t_ini+ "_"+ t_fin + "_" + word +"_";
                        if (verbose){
                                System.out.println("key -> " + new_key);
                        }
                        context.write(new Text(new_key), new LongWritable(1));
                }
                br.close();
        }
} catch (Exception e) {
        e.printStackTrace();
}}}

public static class NewsASRreducer extends Reducer<Text, LongWritable, Text, LongWritable> {
public void reduce(Text key, Iterable<LongWritable> values, Context context)
                        throws IOException, InterruptedException {
```

```
        Configuration conf = context.getConfiguration();
        boolean verbose =  Boolean.parseBoolean(conf.get("verbose.mode"));
        System.out.println("===============  REDUCER   ======================");
        System.out.println("=============== Classname: " +
                                    this.getClass().getName() + "=================");
        long sum = 0;
        for (LongWritable val : values){
        try{
                sum += val.get();
        }catch(Exception e){
                e.printStackTrace();
        }
        }
        if (verbose){
                System.out.println (key + "\t(" + sum + ")");
        }
        context.write(key, new LongWritable(sum));
        }
} // End NewsASRreducer

public static void main(String[] args) throws Exception {
        org.apache.hadoop.conf.Configuration conf = new org.apache.hadoop.conf.Configuration();
        String[] programArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        System.out.println("Total argumentos:" + programArgs.length);
        for (int n = 0; n < programArgs.length; n++) {
                System.out.println(programArgs[n]);
        }
        if (programArgs.length != 2) {
                System.exit(-1);
        }
        String input_file = programArgs[0];
        String output_dir = programArgs[1];
        System.out.println("Input Sequence File:" + input_file);
        System.out.println("Output Dir         :" + output_dir);
        conf.set("output.dir", output_dir);
        conf.set("input_file", input_file);
        conf.set("local.execution", "true");
        conf.set("verbose.mode", "true");
        Job job = new Job(conf);
        job.setJarByClass(NewsASRstage2.class);
        job.setMapperClass(NewsASRmapper.class);
        job.setReducerClass(NewsASRreducer.class);
        job.setInputFormatClass(SequenceFileInputFormat.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LongWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(BytesWritable.class);
        job.setOutputValueClass(LongWritable.class);
        job.setNumReduceTasks(1);
        FileInputFormat.addInputPath(job, new Path(input_file));
        FileOutputFormat.setOutputPath(job, new Path(output_dir));
        job.waitForCompletion(true);
}
}
```

Workflow.xml

Se utiliza de forma conjunta el workflow.xml del Anexo III.

8.5 Anexo V - Ficheros Oozie

newsASRcoordinator.xml

```
<coordinator-app name="newsASR" frequency="${coord:minutes(15)}" start="${start}" end="${end}"
timezone="${timezone}" xmlns="uri:oozie:coordinator:0.1">
        <controls>
                <timeout>${coord:minutes(120)}</timeout>
                <concurrency>1</concurrency>
                <execution>FIFO</execution>
        </controls>
        <datasets>
                <dataset name="din" frequency="${coord:minutes(5)}"
                                initial-instance="${start}" timezone="${timezone}">
                <uri-template>${base_dir}/${YEAR}/${MONTH}/${DAY}/${HOUR}/${MINUTE}</uri-template>
                <done-flag>sequence_file.sf</done-flag>
                </dataset>
                <dataset name="dout" frequency="${coord:minutes(5)}"
                                initial-instance="${start}" timezone="${timezone}">
                <uri-template>${base_dir}/${YEAR}/${MONTH}/${DAY}/${HOUR}/${MINUTE}/output</uri-
template>
                </dataset>
        </datasets>
        <input-events>
                <data-in name="IN1" dataset="din">
                        <instance>${coord:current(0)}</instance>
                </data-in>
        </input-events>
        <output-events>
                <data-out name="OUT" dataset="dout">
                        <instance>${coord:current(0)}</instance>
                </data-out>
        </output-events>
        <action>
        <workflow>
                <app-path>${wf_app_path}</app-path>
                <configuration>
                        <property>
                                <name>sequence_file</name>
                                <value>${coord:dataIn('IN1')}</value>
                        </property>
                        <property>
                                <name>output_dir</name>
                                <value>${coord:dataOut('OUT')}</value>
                        </property>
                        <property>
                                <name>verbose.mode</name>
                                <value>${verbose}</value>
                        </property>
                </configuration>
        </workflow>
        </action>
</coordinator-app>
```

newsASR.properties

```
oozie.coord.application.path=
hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/newsASRcoordinator.xml
frequency=5
min_frequency=5
timezone=UTC
start=2014-12-21T12:00Z
end=2014-12-21T14:00Z
wf_app_path=
hdfs://bivm.ibm.com:9000/user/applications/3bbe5607-be87-4f2a-ba8c-e6ef719a15b5/workflow/workflow.xml
jobTracker=bivm.ibm.com:9001
nameNode=hdfs://bivm.ibm.com:9000
queueName=default
```

```
base_dir=/user/biadmin/newsASR/data

#32MB
#blockSize=33554432

#64MB
#blockSize=67108864

#128MB
blockSize=134217728

#256
#blockSize=268435456

verbose=true
```

createKnow1Coordinator.xml

```
<coordinator-app name="createKnow1" frequency="${coord:minutes(120)}"
                     start="${start}" end="${end}" timezone="${timezone}"
                     xmlns="uri:oozie:coordinator:0.1">
       <controls>
              <timeout>${coord:minutes(120)}</timeout>
              <concurrency>1</concurrency>
              <execution>FIFO</execution>
       </controls>
       <action>
              <workflow>
                     <app-path>${wf_app_path}</app-path>
              </workflow>
       </action>
</coordinator-app>
```

createKnow1.properties

```
oozie.coord.application.path=
hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/createKnow1Coordinator.xml
frequency=15
min_frequency=5
timezone=UTC
start=2014-12-21T12:00Z
end=2014-12-21T12:01Z
wf_app_path=
hdfs://bivm.ibm.com:9000/user/applications/7ea10307-0f8c-4834-8880-02dce42b3966/workflow/workflow.xml
jobTracker=bivm.ibm.com:9001
nameNode=hdfs://bivm.ibm.com:9000
queueName=default
base_dir=/user/biadmin/newsASR/data

#32MB
blockSize=33554432

gap_type=-d
gap=-365
input_file=/user/biadmin/newsASR/temp/filelist.dat
data_dir=/user/biadmin/newsASR/data

output_dir=/user/biadmin/newsASR/temp
stop_word_file=./stopwords
```

createKnow2Coordinator.xml

```
<coordinator-app name="createKnow2" frequency="${coord:minutes(120)}"
               start="${start}" end="${end}" timezone="${timezone}"
               xmlns="uri:oozie:coordinator:0.1">
       <controls>
              <timeout>${coord:minutes(120)}</timeout>
              <concurrency>1</concurrency>
              <execution>FIFO</execution>
       </controls>
       <action>
```

```
                <workflow>
                        <app-path>${wf_app_path}</app-path>
                </workflow>
        </action>
</coordinator-app>
```

createKnow2.properties

```
oozie.coord.application.path=hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/createKnow2Coordinator.xml
frequency=15
min_frequency=5
timezone=UTC
start=2014-12-21T12:00Z
end=2014-12-21T12:01Z
wf_app_path=hdfs://bivm.ibm.com:9000/user/applications/f5067447-fb07-446b-8408-
fa705953d7d5/workflow/workflow.xml
jobTracker=bivm.ibm.com:9001
nameNode=hdfs://bivm.ibm.com:9000
queueName=default
base_dir=/user/biadmin/newsASR/data

#32MB
blockSize=33554432
gap_type=-d
gap=-365
input_file=/user/biadmin/newsASR/temp/filelist.dat
data_dir=/user/biadmin/newsASR/data

output_dir=/user/biadmin/newsASR/temp
stop_word_file=./stopwords
search_word_file=./searchwords
```

createKnow3Coordinator.xml

```
<coordinator-app name="createKnow3" frequency="${coord:minutes(120)}" start="${start}" end="${end}"
timezone="${timezone}" xmlns="uri:oozie:coordinator:0.1">
        <controls>
                <timeout>${coord:minutes(120)}</timeout>
                <concurrency>1</concurrency>
                <execution>FIFO</execution>
        </controls>
        <action>
                <workflow>
                        <app-path>${wf_app_path}</app-path>
                </workflow>
        </action>
</coordinator-app>
```

createKnow3.properties

```
oozie.coord.application.path=hdfs://bivm.ibm.com:9000/user/biadmin/newsASR/run/createKnow3Coordinator.xml
frequency=15
min_frequency=5
timezone=UTC
start=2014-12-21T12:00Z
end=2014-12-21T12:01Z
wf_app_path=hdfs://bivm.ibm.com:9000/user/applications/a4e161b0-d95a-49bb-a13b-
033d7cae1451/workflow/workflow.xml
jobTracker=bivm.ibm.com:9001
nameNode=hdfs://bivm.ibm.com:9000
queueName=default
base_dir=/user/biadmin/newsASR/data

#32MB
blockSize=33554432
gap_type=-d
gap=-365
input_file=/user/biadmin/newsASR/temp/filelist.dat
data_dir=/user/biadmin/newsASR/data
output_dir=/user/biadmin/newsASR/temp
stop_word_file=./stopwords
search_word_file=./searchwords
date_ini=2005/01/01
date_end=2015/12/31
```

8.6 Anexo VI - Generación del Conocimiento: WorkKnow1

genera_file.sh

```
#!/bin/bash

date2stamp () {
    date --utc --date "$1" +%s
    #date --date "$1" +%s
}

dateDiff (){
    #echo $2
    case $1 in
        -s)   sec=1;;
        -m)   sec=60;;
        -h)   sec=3600;;
        -d)   sec=86400;;
        *)    sec=86400;;
    esac
    dte1=$(date2stamp "$2")
    dte2=$(date2stamp "$3")
    diffSec=$((dte2-dte1))
    echo $((diffSec/sec))
}
if [ "$#" -ne 5 ]; then
  echo "Usage : $0 date_start gap_type gap_count dir_search file_exit" >&2
  echo "        date_start  . init time. use now for actual time"
  echo "        gap_type    . -s - seconds"
  echo "                      -m - minutes"
  echo "                      -h - hours"
  echo "                      -d - days"
  echo "        gap_count   . total of gap_type to count"
  echo "        dir_search  . Directory for search files"
  echo "        file_exit   . File to save files founds"
  echo "Sample:       $0 now  -m -15  /user/biadmin/newsASR/data
/users/biadmin/newsASR/temp/list.dat"
  echo "              $0 \"2014-12-24 12:00:00\" -d 19 /user/biadmin/newsASR/data
/users/biadmin/newsASR/temp/list.dat"
  echo "Total parameters: $#"
  exit 1
fi

if [ "$1" == "now" ]; then
     start_time=`date '+%Y-%m-%d %H:%M:00'`
else
     start_time=$1
fi

DIR=$4
TEMPFILE="/tmp/filelist_$$.data"

echo "Start time: "$start_time
echo "Diference :  $3 in $2"
echo "Directory : "$DIR
echo "Generated File: "$5
echo "Temp file : "$TEMPFILE


list=`hadoop fs -ls -R $DIR|awk -F" " '{print $8}'|grep "stage2/part-r-00000"`

for file in $list
do
    #echo $file
    year=`echo $file|awk -F"/" '{print $6}'`
    month=`echo $file|awk -F"/" '{print $7}'`
    day=`echo $file|awk -F"/" '{print $8}'`
    hour=`echo $file|awk -F"/" '{print $9}'`
    min=`echo $file|awk -F"/" '{print $10}'`
    DATE1="$year-$month-$day $hour:$min:00"
    DIFFF=`dateDiff $2 "$start_time" "$DATE1"`
```

```
    DO="DISCARD"
    DIFF=$(($DIFFF-($3)))
    touch $TEMPFILE

    if [ $DIFF -ge 0 ];then
        DO="COPY"
        echo $file >> $TEMPFILE
    fi
    echo "[$start_time - $DATE1 (diff: $DIFFF) $DO ]"$file

done

hadoop fs -put -f $TEMPFILE $5

cat $TEMPFILE
rm $TEMPFILE
```

MapReduce

```
package com.createknow;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

public class JoinWordsFiles {
       public static class JoinWordsFilesMapper extends Mapper<Object, Text, Text, IntWritable> {
              public void map(Object key, Text value, Context context)
                     throws IOException, InterruptedException {
       System.out.println("================================================");
       System.out.println("============ CreateKnow Mapper   ================");
       System.out.println("================================================");
       Configuration conf = context.getConfiguration();
       String output_dir = conf.get("output.dir");
       String input_file = conf.get("mapred.input.dir");
       String stop_words_file=conf.get("stop.words.file");
       System.out.println("Output Dir    : " + output_dir);
       System.out.println("Input File    : " + input_file);
       System.out.println("Stop Word File: " + stop_words_file);
       System.out.println("Value         :" + value);
       try {
              String line = null;
              Path pt=new Path(value.toString());
              FileSystem fs = FileSystem.get(conf);
              BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
              while ((line = br.readLine()) != null) {
                     String word="";
                     String[] b = line.split("_");
                     if (b.length >=6 ) {
                            if (!b[5].toString().isEmpty()){
                                   String [] aux=b[5].toString().split("\\s+");
                                   word=aux[0].trim();
                                   //Deleting stopwords
                                   Scanner sc = new Scanner(new File(stop_words_file));
                                   List<String> lines = new ArrayList<String>();
                                   while (sc.hasNextLine()) {
                                          lines.add(sc.nextLine());
                                   }
```

```
                                String[] stopwords = lines.toArray(new String[0]);
                                //sorting array in java
                                Arrays.sort(stopwords);
                        //searching on sorted array in java using Arrays binarySearch() method
                                if(!(Arrays.binarySearch(stopwords, word) >=0 )){
                                        word = word.trim();
                                        //removing silences
                                        if (!(word.equals("<s>") ||
                                                word.equals("<sil>"))){
                                                context.write (new Text(word),
                                                        new IntWritable(1));
                                        }
                                        }
                                }
                        }

                }
                br.close();
        }catch(Exception e){
                e.printStackTrace();
        }
}}

public static class JoinWordsFilesReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                        throws IOException, InterruptedException {
                System.out.println ("================== REDUCER==================");
                Configuration conf= context.getConfiguration();
                int sum=0;
                for (IntWritable val : values){
                try{
                        sum += val.get();
                }catch(Exception e){
                        e.printStackTrace();
                }
                }
                context.write(key, new IntWritable(sum));
        }
}
public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        //Reading args
        String[] programArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        System.out.println("Total argumentos:" + programArgs.length);
        for (int n = 0; n < programArgs.length; n++) {
                System.out.println(programArgs[n]);
        }

        if (programArgs.length != 3) {
                System.exit(-1);
        }
        String input_file = programArgs[0];
        String output_dir = programArgs[1];
        String stop_words_file=programArgs[2];

        System.out.println("Input File:" + input_file);
        System.out.println("Output Dir         :" + output_dir);
        System.out.println("Stop Words File    :" + stop_words_file);
        conf.set("output.dir", output_dir);
        conf.set("input_file", input_file);
        conf.set("stop.words.file",stop_words_file);
        // Setting job
        Job job = new Job(conf);
        job.setJarByClass(JoinWordsFiles.class);
        job.setMapperClass(JoinWordsFilesMapper.class);
        job.setReducerClass(JoinWordsFilesReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setNumReduceTasks(1);
        conf.set("mapreduce.job.reduces","1");
        conf.set("mapred.reduce.tasks", "1");

        FileInputFormat.addInputPath(job, new Path(input_file));
        FileOutputFormat.setOutputPath(job, new Path(output_dir));
```

```
        System.exit(job.waitForCompletion(true) ? 0 : 1);

}}
```

cloud_generate.R

```
#install.packages("RColorBrewer")
#install.packages("Rcpp")
#install.packages("wordcloud")
message ("WordCloud Generator")

require(RColorBrewer)
require(Rcpp)
require(wordcloud)

# Run system (paste("Rscript c:/rscripts/generate_cloud_r2.r c:/testR/words.txt
c:/rscripts/generate.pdf" ))
# arg1 csv tab separated input file
# arg2 pdf output file in hadoop fs

options(echo=TRUE)                # if you want see commands in output file
args <- commandArgs(trailingOnly = TRUE)

file_input<-args[1]
file_output<-args[2]
file_aux<-"generado_aux.pdf"

message (file_input)
message (file_output)
message (file_aux)

fpe <- read.csv (file=file_input,head=FALSE,sep="\t")

colnames(fpe) <- c("word", "freq")
sorted.fpe <- fpe[order(fpe$freq), ]
n = round(0.2 * nrow(sorted.fpe))
sorted.fpe<-tail(sorted.fpe, n)

pdf(file_aux)

        pal <- brewer.pal(9,"BuGn")
        pal <- pal[-(1:4)]
        wordcloud(fpe$word,fpe$freq,c(8,.3),2,,TRUE,,.15,pal)

        pal <- brewer.pal(6,"Dark2")
        pal <- pal[-(1)]
        wordcloud(fpe$word,fpe$freq,c(8,.3),2,,TRUE,,.15,pal)

        pal <- brewer.pal(9,"BuGn")
        pal <- pal[-(1:4)]
        text(x=0.5, y=0.5, "Plot with best 20% items")
        wordcloud(sorted.fpe$word,sorted.fpe$freq,c(8,.3),2,,TRUE,,.15,pal)

        pal <- brewer.pal(6,"Dark2")
        pal <- pal[-(1)]
        text(x=0.5, y=0.5, "Plot with best 20% items")
        wordcloud(sorted.fpe$word,sorted.fpe$freq,c(8,.3),2,,TRUE,,.15,pal)

dev.off()
command<-paste("hadoop fs -put ",file_aux,file_output)
system(command)
```

workflow.xml

```
<workflow-app name="CreateKnow" xmlns="uri:oozie:workflow:0.1">
<start to="get_file_names_to_process"/>
        <action name='get_file_names_to_process'>
                <shell xmlns="uri:oozie:shell-action:0.1">
                        <job-tracker>${jobTracker}</job-tracker>
                        <name-node>${nameNode}</name-node>
                        <exec>./genera_file.sh</exec>
                        <argument>now</argument>
```

```
                <argument>${gap_type}</argument>
                <argument>${gap}</argument>
                <argument>${data_dir}</argument>
                <argument>${input_file}</argument>
                <file>/user/biadmin/newsASR/lib/genera_file.sh#genera_file.sh</file>
        </shell>
        <ok to="join_files_and_sumarize" />
        <error to="fail" />
</action>

<action name="join_files_and_sumarize">
        <map-reduce>
                <job-tracker>${jobTracker}</job-tracker>
                <name-node>${nameNode}</name-node>
                <prepare>
                        <delete path="${nameNode}/${output_dir}/createknow1"/>
                </prepare>
                <configuration>
                        <property>
                                <name>mapred.job.name</name>
                                <value>CreateKnow</value>
                        </property>
                        <property>
                                <name>mapreduce.map.class</name>
                        <value>com.createknow.JoinWordsFiles$JoinWordsFilesMapper</value>
                        </property>
                        <property>
                                <name>mapreduce.reduce.class</name>
                        <value>com.createknow.JoinWordsFiles$JoinWordsFilesReducer</value>
                        </property>
                        <property>
                                <name>mapred.output.dir</name>
                                <value>${output_dir}/createknow1</value>
                        </property>
                        <property>
                                <name>output.dir</name>
                                <value>${output_dir}/createknow1</value>
                        </property>
                        <property>
                                <name>mapred.input.dir</name>
                                <value>${input_file}</value>
                        </property>
                        <property>
                                <name>stop.words.file</name>
                                <value>${stop_word_file}</value>
                        </property>
                        <property>
                                <name>mapreduce.mapinput.key.class</name>
                                <value>org.apache.hadoop.io.Object</value>
                        </property>
                        <property>
                                <name>mapreduce.mapinput.value.class</name>
                                <value>org.apache.hadoop.io.Text</value>
                        </property>
                        <property>
                                <name>mapred.mapoutput.key.class</name>
                                <value>org.apache.hadoop.io.Text</value>
                        </property>
                        <property>
                                <name>mapred.mapoutput.value.class</name>
                                <value>org.apache.hadoop.io.IntWritable</value>
                        </property>
                        <property>
                                <name>mapred.reducer.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapred.mapper.new-api</name>
                                <value>true</value>
                        </property>
                </configuration>
                <file>/user/biadmin/newsASR/lib/stopwords#stopwords</file>
        </map-reduce>
        <ok to="generate_wordcloud"/>
        <error to="fail"/>
</action>
<action name='generate_wordcloud'>
```

```
            <shell xmlns="uri:oozie:shell-action:0.1">
                  <job-tracker>${jobTracker}</job-tracker>
                  <name-node>${nameNode}</name-node>
                  <exec>/usr/bin/Rscript</exec>
                  <argument>generate.R</argument>
                  <argument>./source.dat</argument>
                  <argument>${output_dir}/createknow1/generate.pdf</argument>
                  <file>${output_dir}/createknow1/part-r-00000#source.dat</file>
                  <file>/user/biadmin/newsASR/lib/cloud_generate.R#generate.R</file>
            </shell>
            <ok to="end" />
            <error to="fail" />
      </action>
      <kill name="fail">
            <message>error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
      </kill>
      <end name="end"/>
</workflow-app>
```

8.7 Anexo VII - Generación del Conocimiento: WorkKnow2

Proceso MapReduce

```
package com.createknow2;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

public class JoinWordsFiles {

public static class JoinWordsFilesMapper extends Mapper<Object, Text, Text, IntWritable> {
public void map(Object key, Text value, Context context)
                                        throws IOException, InterruptedException {
       System.out.println("================================================================");
       System.out.println("======================== CreateKnow2 Mapper   ================");
       System.out.println("================================================================");
       Configuration conf = context.getConfiguration();
       String output_dir = conf.get("output.dir");
       String input_file = conf.get("mapred.input.dir");
       String stop_words_file=conf.get("stop.words.file");
       String search_words_file=conf.get("search.word.file");
       String data_dir=conf.get("data.dir");
       String fichero_procesar=value.toString();
       System.out.println("Output Dir      : " + output_dir);
       System.out.println("Input File      : " + input_file);
       System.out.println("Stop Word File  : " + stop_words_file);
       System.out.println("Value           :" + value);
       System.out.println("Search word file:" + search_words_file);
       System.out.println("Data dir        :" + data_dir);
       System.out.println("Fichero procesar:" + fichero_procesar);
       System.out.println("============================oo==============================");
       try {
              String line = null;
              Path pt=new Path(fichero_procesar);
              FileSystem fs = FileSystem.get(conf);
              BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
              //Reading stopwords file
              Scanner sc = new Scanner(new File(stop_words_file));
              List<String> lines = new ArrayList<String>();
              while (sc.hasNextLine()) {
                     lines.add(sc.nextLine());
              }
              String[] stopwords = lines.toArray(new String[0]);
              //sorting array in java
              Arrays.sort(stopwords);
              Scanner scc= new Scanner(new File(search_words_file));
              List<String> s_words= new ArrayList<String>();
              while (scc.hasNextLine()){
              s_words.add(scc.nextLine());
       }
       String[] f_words = s_words.toArray(new String[0]);
       Arrays.sort(f_words);
```

```
	while ((line = br.readLine()) != null) {
	String word="";
	String[] b = line.split("_");
	if (b.length >=6 ) {
	if (!b[5].toString().isEmpty()){
		String [] aux=b[5].toString().split("\\s+");
		word=aux[0].trim();
		//Deleting stopwords
	//searching on sorted array in java using Arrays binarySearch() method
		if(!(Arrays.binarySearch(stopwords, word) >=0 )){
			word = word.trim();
			//removing silences
			if (!(word.equals("<s>") || word.equals("<sil>"))){
			if((Arrays.binarySearch(f_words, word) >=0 )){
				// Find a word
				System.out.println("Palabra:" + word + "-" + line +"-"+ b[0]);
				System.out.println("-----");
				String aux2= fichero_procesar.replace(data_dir+"/", "");
				System.out.println(aux2);
				String[] split_value = aux2.split("/");
				String anyo=split_value[0];
				String mes=split_value [1];
				String resultado=anyo+""+mes+"01"+ "_"+b[0];
				System.out.println(resultado);
				context.write (new Text(resultado),new IntWritable(1));
			}}
		}
	}
	}
	}
	br.close();
	}catch(Exception e){
		e.printStackTrace();
	}
}}

public static class JoinWordsFilesReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context)
                                                    throws IOException, InterruptedException {
	System.out.println ("================ REDUCER======================");
	Configuration conf= context.getConfiguration();
	int sum=0;
	for (IntWritable val : values){
		try{
			sum += val.get();
		}catch(Exception e){
			e.printStackTrace();
		}
	}
	context.write(key, new IntWritable(sum));
}
}
public static void main(String[] args) throws Exception {
	Configuration conf = new Configuration();
	//Reading args
	String[] programArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
	System.out.println("Total argumentos:" + programArgs.length);
	for (int n = 0; n < programArgs.length; n++) {
		System.out.println(programArgs[n]);
	}
	if (programArgs.length != 4) {
		System.exit(-1);
	}
	String input_file = programArgs[0];
	String output_dir = programArgs[1];
	String stop_words_file=programArgs[2];
	String search_words_file=programArgs[3];
	System.out.println("Input File:" + input_file);
	System.out.println("Output Dir         :" + output_dir);
	System.out.println("Stop Words File    :" + stop_words_file);
	conf.set("output.dir", output_dir);
	conf.set("input_file", input_file);
	conf.set("stop.words.file",stop_words_file);
	conf.set("search.word.file",search_words_file);
	conf.set("data.dir", "/home/joseh/workspace/CreateKnow2/test/data");
	// Setting job
	Job job = new Job(conf);
```

```
        job.setJarByClass(JoinWordsFiles.class);
        job.setMapperClass(JoinWordsFilesMapper.class);
        job.setReducerClass(JoinWordsFilesReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setNumReduceTasks(1);
        conf.set("mapreduce.job.reduces","1");
        conf.set("mapred.reduce.tasks", "1");
        FileInputFormat.addInputPath(job, new Path(input_file));
        FileOutputFormat.setOutputPath(job, new Path(output_dir));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

generate2.R

```
require("ggplot2")

# Run system (paste("Rscript c:/rscripts/generate_cloud_r2.r c:/testR/words.txt
c:/rscripts/generate.pdf" ))
# arg1 csv tab separated input file
# arg2 pdf output file in hadoop fs

options(echo=TRUE)                          # if you want see commands in output file
args <- commandArgs(trailingOnly = TRUE)

file_input<-args[1]
file_output<-args[2]
file_aux<-"generado_aux.pdf"

message (file_input)
message (file_output)
message (file_aux)

fpe <- read.csv (file=file_input,head=FALSE,sep="\t")
fpe$anyo<- substr(fpe$V1,1,4)
fpe$mes<-substr(fpe$V1,5,6)
fpe$dia<-substr(fpe$V1,7,8)

fpe$word <-  do.call(rbind, strsplit(as.character(fpe$V1), '_'))[,c(2)]
fpe$anyo_mes <-  as.numeric(fpe$anyo)*100+as.numeric(fpe$mes)
fpe$anyo_mes2 <-  as.Date(paste(fpe$anyo,"-",fpe$mes,"-","01" , sep = ""))

pdf(file_aux)

sleepplot<-ggplot(fpe, aes(y = V2, x = anyo_mes2, colour = word))
sleepplot <- sleepplot +  geom_point() + geom_line()
sleepplot <- sleepplot + geom_smooth(span = 0.5, fill = NA)
sleepplot <- sleepplot + xlab("Fecha") + ylab("Apariciones") + ggtitle("Agrupaciones de palabras")
sleepplot <- sleepplot + facet_grid(word~.)
sleepplot <- sleepplot + theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(sleepplot)

sleepplot<-ggplot(fpe, aes(y = V2, x = anyo_mes2, colour = word))
sleepplot <- sleepplot +  geom_point() + geom_line()
sleepplot <- sleepplot + geom_smooth(span = 0.5, fill = NA)
sleepplot <- sleepplot + xlab("Fecha") + ylab("Apariciones") + ggtitle("Agrupaciones de palabras")
sleepplot <- sleepplot + facet_grid(.~word)
sleepplot <- sleepplot + theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(sleepplot)

sleepplot<-ggplot(fpe, aes(y = V2, x = anyo_mes2, colour = word))
sleepplot <- sleepplot +  geom_point() + geom_line()
sleepplot <- sleepplot + geom_smooth(span = 0.5, fill = NA)
sleepplot <- sleepplot + xlab("Fecha") + ylab("Apariciones") + ggtitle("Agrupaciones de palabras")
sleepplot <- sleepplot + theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(sleepplot)
dev.off()
command<-paste("hadoop fs -put ",file_aux,file_output)
command

system("ls -la")
system(command)
```

workflow.xml

```
<workflow-app name="CreateKnow2" xmlns="uri:oozie:workflow:0.1">
<start to="get_file_names_to_process"/>
	<action name='get_file_names_to_process'>
		<shell xmlns="uri:oozie:shell-action:0.1">
			<job-tracker>${jobTracker}</job-tracker>
			<name-node>${nameNode}</name-node>
			<exec>./genera_file.sh</exec>
			<argument>now</argument>
			<argument>${gap_type}</argument>
			<argument>${gap}</argument>
			<argument>${data_dir}</argument>
			<argument>${input_file}</argument>
			<file>/user/biadmin/newsASR/lib/genera_file.sh#genera_file.sh</file>
		</shell>
		<ok to="join_files_and_sumarize" />
		<error to="fail" />
	</action>
	<action name="join_files_and_sumarize">
		<map-reduce>
			<job-tracker>${jobTracker}</job-tracker>
			<name-node>${nameNode}</name-node>
			<prepare>
				<delete path="${nameNode}/${output_dir}/createknow2"/>
			</prepare>
			<configuration>
				<property>
					<name>mapred.job.name</name>
					<value>CreateKnow2</value>
				</property>
				<property>
					<name>mapreduce.map.class</name>
				<value>com.createknow2.JoinWordsFiles$JoinWordsFilesMapper</value>
				</property>
				<property>
					<name>mapreduce.reduce.class</name>
				<value>com.createknow2.JoinWordsFiles$JoinWordsFilesReducer</value>
				</property>
				<property>
					<name>mapred.output.dir</name>
					<value>${output_dir}/createknow2</value>
				</property>
				<property>
					<name>output.dir</name>
					<value>${output_dir}/createknow2</value>
				</property>
				<property>
					<name>mapred.input.dir</name>
					<value>${input_file}</value>
				</property>
				<property>
					<name>stop.words.file</name>
					<value>${stop_word_file}</value>
				</property>
				<property>
					<name>search.word.file</name>
					<value>${search_word_file}</value>
				</property>
				<property>
					<name>data.dir</name>
					<value>${data_dir}</value>
				</property>
				<property>
					<name>mapreduce.mapinput.key.class</name>
					<value>org.apache.hadoop.io.Object</value>
				</property>
				<property>
					<name>mapreduce.mapinput.value.class</name>
					<value>org.apache.hadoop.io.Text</value>
				</property>
				<property>
					<name>mapred.mapoutput.key.class</name>
					<value>org.apache.hadoop.io.Text</value>
				</property>
				<property>
```

```
                                <name>mapred.mapoutput.value.class</name>
                                <value>org.apache.hadoop.io.IntWritable</value>
                        </property>
                        <property>
                                <name>mapred.reducer.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapred.mapper.new-api</name>
                                <value>true</value>
                        </property>
                </configuration>
                <file>/user/biadmin/newsASR/lib/selectwords#searchwords</file>
                <file>/user/biadmin/newsASR/lib/stopwords#stopwords</file>
        </map-reduce>
        <ok to="generate_plot2"/>
        <error to="fail"/>
    </action>
    <action name='generate_plot2'>
        <shell xmlns="uri:oozie:shell-action:0.1">
                <job-tracker>${jobTracker}</job-tracker>
                <name-node>${nameNode}</name-node>
                <exec>/usr/bin/Rscript</exec>
                <argument>generate2.R</argument>
                <argument>./source.dat</argument>
                <argument>${output_dir}/createknow2/generate.pdf</argument>
                <file>${output_dir}/createknow2/part-r-00000#source.dat</file>
                <file>/user/biadmin/newsASR/lib/graph_generate2.R#generate2.R</file>
        </shell>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

8.8 Anexo VIII - Generación del Conocimiento: WorkKnow3

Proceso MapReduce

```
package com.createknow3;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.Scanner;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

public class JoinWordsFiles {
       public static class JoinWordsFilesMapper extends Mapper<Object, Text, Text, IntWritable> {
       public void map(Object key, Text value, Context context) throws IOException,
              InterruptedException {
              System.out.println("================================================");
              System.out.println("============== CreateKnow3 Mapper   =============");
              System.out.println("================================================");
              Configuration conf = context.getConfiguration();
              String output_dir = conf.get("output.dir");
              String input_file = conf.get("mapred.input.dir");
              String stop_words_file=conf.get("stop.words.file");
              String search_words_file=conf.get("search.word.file");
              String data_dir=conf.get("data.dir");
              String date_ini=conf.get("date.ini");
              String date_end=conf.get("date.end");
              String fichero_procesar=value.toString();
              System.out.println("Output Dir      : " + output_dir);
              System.out.println("Input File      : " + input_file);
              System.out.println("Stop Word File  : " + stop_words_file);
              System.out.println("Value           :" + value);
              System.out.println("Search word file:" + search_words_file);
              System.out.println("Data dir        :" + data_dir);
              System.out.println("Fichero procesar:" + fichero_procesar);
              System.out.println("date ini        :" + date_ini);
              System.out.println("date end        :" + date_end);
              try {
                     String aux3= fichero_procesar.replace(data_dir+"/", "");
                     System.out.println(aux3);
                     String[] split_value2 = aux3.split("/");
                     String anyo1=split_value2[0];
                     String mes1=split_value2 [1];
                     String dia1=split_value2 [2];
                     DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
                     Date d_ini = dateFormat.parse(date_ini);
                     System.out.println("Date ini : " + d_ini);
                     Date d_end = dateFormat.parse(date_end);
                     System.out.println("Date end : " + d_end);
                     Date d_file = dateFormat.parse(anyo1+"/"+mes1+"/"+dia1);
                     System.out.println("Date file : " + d_file);
```

```
                        if (d_ini.compareTo(d_file)<=0 && d_end.compareTo(d_file)>=0){
                                System.out.println("Fichero aceptado entre las fechas de selección");
                                String line = null;
                                Path pt=new Path(fichero_procesar);
                                FileSystem fs = FileSystem.get(conf);
                                BufferedReader br
                                =new BufferedReader(new InputStreamReader(fs.open(pt)));
                                Scanner sc = new Scanner(new File(stop_words_file));
                                List<String> lines = new ArrayList<String>();
                                while (sc.hasNextLine()) {
                                        lines.add(sc.nextLine());
                                }
                                String[] stopwords = lines.toArray(new String[0]);
                                //sorting array in java
                                Arrays.sort(stopwords);
                                Scanner scc= new Scanner(new File(search_words_file));
                                List<String> s_words= new ArrayList<String>();
                                while (scc.hasNextLine()){
                                        s_words.add(scc.nextLine());
                                }
                                String[] f_words = s_words.toArray(new String[0]);
                                Arrays.sort(f_words);
                                while ((line = br.readLine()) != null) {
                                        String word="";
                                        String[] b = line.split("_");
                                        if (b.length >=6 ) {
                                                if (!b[5].toString().isEmpty()){
                                                        String [] aux=b[5].toString().split("\\s+");
                                                        word=aux[0].trim();
                                                if(!(Arrays.binarySearch(stopwords, word) >=0 )){
                                                        word = word.trim();
                                                if (!(word.equals("<s>") || word.equals("<sil>"))){
                                                if((Arrays.binarySearch(f_words, word) >=0 )){
                                                // Find a word
                                                System.out.println("Palabra:" + word
                                                                        + "-" + line +"-"+ b[0]);
                                                String aux2
                                                = fichero_procesar.replace(data_dir+"/", "");
                                                String[] split_value = aux2.split("/");
                                                String anyo=split_value[0];
                                                String mes=split_value [1];
                                                String resultado=anyo+""+mes+"01"+ "_"+b[0];
                                                context.write (new Text(resultado),
                                                                new IntWritable(1));
                                                }
                                                }
                                                }
                                        }
                                }
                                }
                                br.close();
                        }else{
                                System.out.println("Fichero NO aceptado.");
                        }
                        }catch(Exception e){
                                e.printStackTrace();
                        }
        }
}
public static class JoinWordsFilesReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
        System.out.println ("============ REDUCER createknow3=================");
        Configuration conf= context.getConfiguration();
        int sum=0;
        for (IntWritable val : values){
                try{
                        sum += val.get();
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
        context.write(key, new IntWritable(sum));
}
}
public static void main(String[] args) throws Exception {
```

```
        Configuration conf = new Configuration();
        String[] programArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        System.out.println("Total argumentos:" + programArgs.length);
        for (int n = 0; n < programArgs.length; n++) {
                System.out.println(programArgs[n]);
        }
        if (programArgs.length != 4) {
                System.exit(-1);
        }
        String input_file = programArgs[0];
        String output_dir = programArgs[1];
        String stop_words_file=programArgs[2];
        String search_words_file=programArgs[3];
        System.out.println("Input File:" + input_file);
        System.out.println("Output Dir         :" + output_dir);
        System.out.println("Stop Words File    :" + stop_words_file);
        conf.set("output.dir", output_dir);
        conf.set("input_file", input_file);
        conf.set("stop.words.file",stop_words_file);
        conf.set("search.word.file",search_words_file);
        conf.set("data.dir", "/home/joseh/workspace/CreateKnow3/test/data");
        conf.set("date.ini","2013/01/01");
        conf.set("date.end","2015/12/31");
        Job job = new Job(conf);
        job.setJarByClass(JoinWordsFiles.class);
        job.setMapperClass(JoinWordsFilesMapper.class);
        job.setReducerClass(JoinWordsFilesReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setNumReduceTasks(1);
        conf.set("mapreduce.job.reduces","1");
        conf.set("mapred.reduce.tasks", "1");
        FileInputFormat.addInputPath(job, new Path(input_file));
        FileOutputFormat.setOutputPath(job, new Path(output_dir));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

generate3.R

```
#install.packages("reshape")
message ("CreateKnow3 Generation")
require(reshape)
require(ggplot2)
require(scales)
# arg1 csv tab separated input file
# arg2 pdf output file in hadoop fs
options(echo=TRUE)                              # if you want see commands in output file
args <- commandArgs(trailingOnly = TRUE)
file_input<-args[1]
file_output<-args[2]
file_aux<-"generado3_aux.pdf"
message (file_input)
message (file_output)
message (file_aux)
#######################################################
####    HEAT MAP
#######################################################
pdf(file_aux)
        fpe <- read.csv (file=file_input,head=FALSE,sep="\t")
        fpe$anyo<- substr(do.call(rbind, strsplit(as.character(fpe$V1), '_'))[,c(1)], 1, 4)
        fpe$mes<- substr(do.call(rbind, strsplit(as.character(fpe$V1), '_'))[,c(1)], 5, 6)
        fpe$V1<-NULL
        fpe$V3<-as.numeric(fpe$V2)
        fpe$V2<-NULL
        aa <-       ggplot(fpe, aes(x=mes, y=anyo, fill=V3))
        aa <- aa  + geom_tile()
        aa <- aa  + scale_fill_gradient2(high="red",mid="white",low="blue")
        aa <- aa  + xlab("Months") + ylab("Year") + ggtitle("Total occurrences aggregation")
        print(aa)
dev.off()
command<-paste("hadoop fs -put ",file_aux,file_output)
command
system("ls -la")
system(command)
```

workflow.xml

```
<workflow-app name="CreateKnow3" xmlns="uri:oozie:workflow:0.1">
	<start to="sumarize_and_join"/>
	<action name="sumarize_and_join">
		<map-reduce>
			<job-tracker>${jobTracker}</job-tracker>
			<name-node>${nameNode}</name-node>
			<prepare>
				<delete path="${nameNode}/${output_dir}/createknow3"/>
			</prepare>
			<configuration>
				<property>
					<name>mapred.job.name</name>
					<value>CreateKnow3</value>
				</property>
				<property>
					<name>mapreduce.map.class</name>
				<value>com.createknow3.JoinWordsFiles$JoinWordsFilesMapper</value>
				</property>
				<property>
					<name>mapreduce.reduce.class</name>
				<value>com.createknow3.JoinWordsFiles$JoinWordsFilesReducer</value>
				</property>
				<property>
					<name>mapred.output.dir</name>
					<value>${output_dir}/createknow3</value>
				</property>
				<property>
					<name>output.dir</name>
					<value>${output_dir}/createknow3</value>
				</property>
				<property>
					<name>mapred.input.dir</name>
					<value>${input_file}</value>
				</property>
				<property>
					<name>stop.words.file</name>
					<value>${stop_word_file}</value>
				</property>
				<property>
					<name>search.word.file</name>
					<value>${search_word_file}</value>
				</property>
				<property>
					<name>date.ini</name>
					<value>${date_ini}</value>
				</property>
				<property>
					<name>date.end</name>
					<value>${date_end}</value>
				</property>
				<property>
					<name>data.dir</name>
					<value>${data_dir}</value>
				</property>
				<property>
					<name>mapreduce.mapinput.key.class</name>
					<value>org.apache.hadoop.io.Object</value>
				</property>
				<property>
					<name>mapreduce.mapinput.value.class</name>
					<value>org.apache.hadoop.io.Text</value>
				</property>
				<property>
					<name>mapred.mapoutput.key.class</name>
					<value>org.apache.hadoop.io.Text</value>
				</property>
				<property>
					<name>mapred.mapoutput.value.class</name>
					<value>org.apache.hadoop.io.IntWritable</value>
				</property>
```

```
                        <property>
                                <name>mapred.reducer.new-api</name>
                                <value>true</value>
                        </property>
                        <property>
                                <name>mapred.mapper.new-api</name>
                                <value>true</value>
                        </property>
                </configuration>
                <file>/user/biadmin/newsASR/lib/selectwords#searchwords</file>
                <file>/user/biadmin/newsASR/lib/stopwords#stopwords</file>
        </map-reduce>
        <ok to="generate_plot3"/>
        <error to="fail"/>
    </action>
    <action name='generate_plot3'>
        <shell xmlns="uri:oozie:shell-action:0.1">
                <job-tracker>${jobTracker}</job-tracker>
                <name-node>${nameNode}</name-node>
                <exec>/usr/bin/Rscript</exec>
                <argument>generate3.R</argument>
                <argument>./source.dat</argument>
                <argument>${output_dir}/createknow3/generate.pdf</argument>
                <file>${output_dir}/createknow3/part-r-00000#source.dat</file>
                <file>/user/biadmin/newsASR/lib/graph_generate3.R#generate3.R</file>
        </shell>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

8.9 Anexo IX - Ejemplos y muestras de ficheros utilizados

Fragmento del fichero resultado del reconocicmiento

```
radio-test-10_000789_000541_00000490_00000840_<s>_  1
radio-test-10_000789_000541_00000840_00001360_el_   1
radio-test-10_000789_000541_00001360_00001690_gozne_       1
radio-test-10_000789_000541_00001690_00001820_es_   1
radio-test-10_000789_000541_00001820_00002180_<sil>_       1
radio-test-10_000789_000541_00002180_00002430_oído_ 1
radio-test-10_000789_000541_00002430_00002880_<sil>_       1
radio-test-10_000789_000541_00002880_00002970_resplandor_  1
radio-test-10_000789_000541_00002970_00003330_de_   1
radio-test-10_000789_000541_00003330_00003620_ojeda_       1
radio-test-10_000789_000541_00003620_00003820_sus_  1
radio-test-10_000789_000541_00003820_00003910_una_  1
radio-test-10_000789_000541_00003910_00003911_<sil>_       1
```

Fragmento del fichero de ejemplo de cabecera del fichero sequence

```
SEQ□□org.apache.hadoop.io.Text"org.apache.hadoop.io.BytesWritable□□*org.apache.hadoop.io.compress.
DefaultCodec□:aa□□□□N□]□□□□□□:aa□□□□N□]□□    x□□□□□□x□□,JL□□□-I-.□54□
```

Fragmento del fichero filelist

```
/user/biadmin/newsASR/data/2013/02/01/12/00/output/stage2/part-r-00000
/user/biadmin/newsASR/data/2013/03/01/12/00/output/stage2/part-r-00000
/user/biadmin/newsASR/data/2013/04/01/12/00/output/stage2/part-r-00000
/user/biadmin/newsASR/data/2013/05/01/12/00/output/stage2/part-r-00000
/user/biadmin/newsASR/data/2013/06/01/12/00/output/stage2/part-r-00000
```

Fragmento del fichero de *stopwords*

```
ninguno
ningunos
no
nos
nosotras
nosotros
nuestra
nuestras
nuestro
nuestros
nunca
o
os
otra
otras
otro
otros
para
parecer
pero
```

Fragmento del fichero de selección (selectwords)

```
azul
ojeda
```