UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

Topic Modeling y Big Data

Bruno Stampete

Junio 2016

Topic Modeling y Big Data

AUTOR: Bruno Stampete
TUTOR: Estrella Pulido Cañabate

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2016

Resumen

El objetivo de este proyecto es diseñar un sistema capaz de inferir una estructura latente de temas, presente en una gran colección de documentos procedentes de la prensa y en idioma castellano. Para conseguir dicho propósito se implementan diferentes variantes del algoritmo Latent Dirichlet Allocation, en concreto, haciendo uso de inferencia bayesiana variacional. Dicho análisis incluye pruebas en entornos distribuidos como el proporcionado por los frameworks Apache Hadoop y Apache Spark.

Tras las primeras pruebas, se evalúa el sistema que mejor se adapta a los objetivos del presente proyecto y se implementa un sistema completo de separación de palabras en tokens, pre-filtrado de los mismos, creación de diccionarios y generación de modelos LDA parametrizando las variables más significativas. Este primer desarrollo se hace sobre todo el corpus.

En un segundo lugar, se aplica el mismo análisis a las distintas secciones de las que se compone el corpus, que ha sido clasificado de forma manual previamente. El objetivo de este segundo análisis es poder inferir la estructura temática que subyace en cada tema de la colección de noticias de prensa y, consiguientemente, poder comparar la semejanza que existe entre ficheros externos al corpus de entrenamiento y dichas estructuras.

Por último, se plantea un proceso de optimización de las variables parametrizadas, empleando una validación cruzada, para determinar los valores de estas últimas que maximicen la tasa de acierto por cada tema.

Palabras Claves

Topic Modeling, Big Data, Generación de estructuras de modelos temáticos, Latent Dirichlet Allocation, Apache Hadoop, Apache Spark, Procesado distribuido, Clasificación de prensa.

Abstract

The goal of this project was the implementation of a Topic Modeling system using Latent Dirichlet Allocation over a distributed platform. This analysis has been performed using a Spanish corpus, discovering the themes that pervade a large collection of pieces of news, from El País newspaper. To be able to perform this analysis, several distributed frameworks are taken into consideration, like Apache Hadoop and Apache Spark.

Once implemented, the system is applied to classified parts of the corpus, being able to create different thematic structures. These models will be one per each category of the corpus and the goal of this section would be to compare files not included in the training corpus with these structures, in order to find out the relation between the category and the latent structure, obtained from the document.

Finally, an optimization process is conducted to optimize the parametric variables of the system to maximize the hit rate, using a cross-validation system, and assessing the result for each one of the categories of the corpus.

Key Words:

Topic Modeling, Big Data, Generation of Topic models, Latent Dirichlet Allocation, Apache Hadoop, Apache Spark, Distributed processing, Press news classification.

En memoria de mi padre,
Nullius boni sine socio iucunda possessio est.
Séneca, Ep. ad Lucilium VI

Agradecimientos

En primer lugar, quería agradecer de corazón a mi tutora Estrella Pulido, que en un momento tan delicado de mi vida, ha depositado en mí toda su confianza y me ha permitido realizar un proyecto que suscitara mi interés, respetando mis tiempos y mis obligaciones profesionales.

También quería mostrar mi total agradecimiento a mi jefe, Daniel Tapias, que me ha permitido desarrollar el proyecto en su empresa, usando recursos de la misma y entendiendo en todo momento mi necesidad de compatibilizar la jornada laboral con mi vida de estudiante.

Un agradecimiento muy especial también a Javier Antón, por ser mi fiel amigo, mi compañero de andanzas en la universidad, en el trabajo y en la vida personal, en los buenos y en los malos momentos. Muchísimas gracias por todo el apoyo que he recibido, te seré siempre grato.

Querría agradecer a Juan Manuel Perero por ayudarme en las fases más difíciles del proyecto, por escucharme cuando me hacía falta repasar en voz alta los conceptos o por ayudarme con las temibles expresiones regulares.
Un sincero agradecimiento a Jorge Rico y a Carolina Camacho por todos los consejos y la ayuda recibida en este último año transcurrido juntos en el departamento de I+D.

Deseo también dar las gracias a todos mis compañeros de trabajo, por haber hecho más ameno este periodo en el que me he enfrentado a la ardua tarea de compaginar el trabajo con mi proyecto.

Así mismo, quiero dedicar parte de este proyecto a mis compañeros de la carrera con los que he compartido momentos inolvidables de mi vida de estudiante. Alicia, Almudena, Eva, Raquel, Tamara, Gustavo, Miguel, Rosely, los Álvaros y los Sergios, os doy las gracias, porque con vosotros he vivido las mejores experiencias de mi vida y sin vuestra ayuda no hubiera podido llegar a culminar esta etapa.

Gracias también a ti, Jon, por todo el cariño que me has proporcionado y por haberme apoyado incondicionalmente a lo largo de este periodo tan difícil para mi. Tu presencia y sosiego han sido de vital importancia para poder seguir adelante.

Me gustaría poder compartir este momento, también, con todos mis amigos, que me han animado a seguir adelante, demostrándome en repetidas ocasiones que la distancia no es ninguna barrera. Por todo el apoyo y cariño recibido, Carmen, Chiara, Elena, M. Chiara, Paula, Sergi, Vaidas: Grazie Mille, Tack så mycket!

Por último, pero no por eso menos importante, quería mostrar mi más profundo sentimiento de agradecimiento hacia mis padres, que lo han entregado todo para que no me faltara nunca nada, que me han apoyado en todo momento a lo largo de mis estudios y que nunca han dejado de creer en mi. A mi madre por enseñarme el Know How de la vida y a mi padre por encender en mi la pasión por la ciencia y la ingeniería, les estoy eternamente agradecido.

Bruno Stampete
2 de Junio de 2016

Índice General

Índice de Figuras

Índice de Tablas

Acrónimos

ART	Author-Recipient Topic
AWS	Amazon Web Services
BI	Business Intelligence
DAG	Directed Acyclic Graph
DF	Data Frame
EC2	Elastic Compute Cloud
EM	Expectation Maximization Algorithm
EP	Expectation Propagation
ETL	Extract Tranform and Load
HDFS	Hadoop Distributed File System
JSON	JavaScript Object Notation
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
MLlib	Apache Spark Machine Learning Library
Mr.LDA	MapReduce Latent Dirichlet Allocation
PGM	Probabilistic Graphical Model
pLSA	Probabilistic Latent Semantic Analysis
RDD	Resilent Distributed dataset
TF-IDF	Term frequency/Inverse document frequency
SVD	Singular Value Decomposition
VB	Variational Bayesian
WASB	Azure Storage Blobs
YARN	Yet Another Resource Negotiator

Capítulo 1

Introducción

1.1 Motivación

El Big Data ha tenido un rápido desarrollo en la última década, en gran medida debido a la necesidad de procesar grandes conjuntos de información, que de otra forma no sería posible. El vertiginoso aumento en la cantidad de datos que nuestra sociedad genera, está principalmente ligado a la movilidad y conectividad de los dispositivos, que han cambiado radicalmente la forma en la que nos relacionamos con nuestro entorno.

Para poder procesar dicho volumen de información, se ha hecho necesario desarrollar una serie de herramientas capaces de trabajar en entornos distribuidos con las que los datos se procesan en paralelo, en clústeres de servidores. Dichos frameworks permiten extrapolar y transformar grandes cantidades de datos inclusive en tiempo real, por lo que las aplicaciones que se pueden diseñar usando este enfoque son infinitas.

El Topic Modeling es una de estas herramientas de análisis que permite encontrar los temas sobre los que trata una colección de documentos y de forma más genérica infiere las relaciones entre las palabras, basándose en modelos ocultos de frecuencia de las mismas. Este análisis permite entender un corpus a través de las vocablos que lo componen, sin tener que realizar un trabajo manual de etiquetado.

Por lo que, la motivación del presente proyecto de fin de carrera será estudiar el funcionamiento de un sistema que implemente Topic Modeling sobre plataformas

distribuidas haciendo uso de Big Data y ver las posibles aplicaciones que se pueden crear a partir de dicho desarrollo.

Dado que la clasificación de noticias, notas de prensa y transcripciones de vídeo en el ámbito de los medios de comunicación, sigue siendo una tarea que se realiza a mano, una de las posibles aplicaciones que se quiere estudiar es relacionar las etiquetas con los temas en los que se subdivide el corpus tras haber aplicado un análisis de Topic Modeling.

1.2 Objetivos

El objetivo principal de este proyecto es poder crear un sistema de inferencia de estructuras latentes de temas, sobre grandes colecciones de datos en un entorno distribuido.

Para poder alcanzar dicho objetivo se han realizado pruebas en diferentes entornos de programación, haciendo uso de herramientas y frameworks distintos:

- Mallet [1], para unas primeras pruebas en entorno no distribuido.
- Apache Hadoop [2], para poder familiarizarse con la programación distribuida y los conceptos de Topic Modeling.
- Apache Spark [3], para una implementación funcional y definitiva del sistema.

Una vez obtenidos los primeros resultados, se ha creado un sistema capaz de generar modelos distintos a partir de fracciones del corpus, previamente clasificadas. Con dicho sistema, se puede analizar la semejanza de documentos externos al corpus de entrenamiento con respecto a los temas que componen cada una de las estructuras generadas.

El objetivo final de este proyecto es mostrar el grado de semejanza entre la estructura latente de un documento, no perteneciente al corpus de entrenamiento, y los modelos creados previamente, demostrando que existe una relación entre la estructura de mayor similitud y la categoría del documento.

1.3 Estructura de la memoria

La presente memoria está dividida en los siguientes capítulos:

- **Capítulo 1. Introducción: motivación y objetivos del proyecto.**
  En este mismo capítulo se explican brevemente las motivaciones que han llevado a la realización de este proyecto y los objetivos que se persiguen.

- **Capítulo 2. Estado del Arte: Big Data, Topic Modeling y Frameworks.**
  En este capítulo, dedicado al estado del arte de la tecnología empleada en la realización de este proyecto, se introducirá el concepto de Big Data y de Topic Modeling, haciendo hincapié en el algoritmo Latent Dirichlet Allocation (LDA) [17]. Por último, se analizarán las principales herramientas para poder realizar dicho análisis sobre sistemas distribuidos.

- **Capítulo 3. Recursos: corpus y servidor.**
  En este capítulo se describirán brevemente el origen del corpus empleado para el análisis y las características del servidor que se ha utilizado para la realización de las pruebas.

- **Capítulo 4. Generación de modelos.**
  En este capítulo se detallan los pasos seguidos para crear un sistema capaz de generar modelos LDA de estructuras temáticas en diferentes plataformas, detallando el funcionamiento de los diferentes módulos para la implementación en Scala sobre Apache Spark.

- **Capítulo 5. Generación de modelos temáticos y optimización.**
  En este capítulo se plantea desarrollar un sistema que genere diferentes modelos LDA, cada uno entrenado con una diferente sección del corpus. Posteriormente, se evaluará la semejanza de textos no pertenecientes al corpus de entrenamiento con los modelos generados previamente.

- **Capítulo 6. Conclusiones y trabajo futuros.**
  En este último capítulo se describen los resultados obtenidos a lo largo del desarrollo de este proyecto de fin de carrera y se plantean las líneas futuras de desarrollo que pueden derivar de este estudio.

- **Bibliografía y Anexos.**
  En esta sección se enumeran las referencias bibliográficas y todos los documentos, figuras y tablas que, por cuestión de espacio, no han tenido cabida en los capítulos anteriores.

Capítulo 2

Estado del Arte

En este capítulo se hará una breve introducción al análisis de Big Data (sección 2.1) y posteriormente se detallará el funcionamiento de los algoritmos de Topic Modeling (sección 2.2) haciendo especial hincapié en el algoritmo de Latent Dirichlet Allocation LDA (sección 2.2.2). A continuación se introducirán los Frameworks con los que se puede realizar dicho análisis (sección 2.3).

2.1 Introducción general al análisis de Big Data

2.1.1 Sociedad de la información

El ser humano ha desarrollado una innata capacidad para analizar los estímulos que recibe de su entorno, procesarlos y transformarlos en información. La escritura fue un gran hito para la humanidad, ya que permitió poder guardar el fruto de esos procesos lógicos y cognitivos, creando la posibilidad de utilizar estos datos como punto de partida para otros razonamientos y añadir complejidad a los mismos.

Este proceso fue lento, al comienzo, pero imparable, de amplia difusión y sobre todo de carácter exponencial, por lo que la información que el hombre ha ido creando a lo largo de la historia alcanza cifras exorbitantes. Se estima que toda la información creada por la humanidad antes del siglo XXI en forma de soporte analógico alcanza los 5 Exabytes (5.000 millones de Gigabytes) [4].

Otro gran punto de inflexión fue la llegada de la computación que ha acelerado, definitivamente, tanto el proceso de creación de datos como el aumento de la capacidad de almacenamiento de éstos. Según un estudio de la Universidad de

Berkeley [5], el año 1996 fue clave, ya que el coste de guardar información de forma analógica, en papel por ejemplo, había superado el coste de guardar la información en forma digital. Hubo que esperar hasta 2002 para que la producción de nueva información en formato analógico empezara a disminuir. Estos hechos hicieron que a lo largo de los últimos años del siglo XX y los primeros años de este siglo el aumento de información digital alcanzase un 30-40% anual. Sólo en el año 1999 se crearon unos 2 Exabytes de información, mientras que en 2002 se produjeron en el mundo 5 Exabytes de información [6], lo mismo que se había generado en los 5 milenios anteriores en formato analógico.

El ritmo de crecimiento de datos, como se ha visto, es exponencial y cada vez vemos cómo nuevas tecnologías permiten procesar y almacenar más cantidad de información con costes menores. Además, a partir de 2007 con la aparición de los primeros smartphones se ha visto cómo la movilidad de estos dispositivos otorga más profundidad a los datos añadiendo valor a los mismos. Estos factores han hecho necesario el estudio de grandes cantidades de datos y la creación de herramientas que fueran capaces de operar con ellos. A finales de 2008 la cantidad de información procesada había superado los 9.000 Exabytes pero la mayoría de esta información seguía siendo generada por servidores y no era información de carácter público [7].

En los últimos años, con la gran difusión de los dispositivos móviles, la consolidación de grandes motores de búsqueda como Google o Yahoo! y la aparición de las redes sociales, se ha puesto el foco de atención sobre la cantidad de información que se produce en internet, concretamente el tráfico IP, que cada vez crece más y se ha convertido en el nuevo punto de atención de los analistas.

En pocos años la unidad de medida, el Exabyte, ya no es suficiente para medir la cantidad de información generada por el ser humano y de ahí que Cisco emplee el término '**The Zettabyte Era'** [8] ya que actualmente la información que circula en un año a través del protocolo IP ha alcanzado el Zettabyte (1.000 Exabytes) y empresas como IDC pronostican que para el 2020 se producirán unos 40 Zettabytes al año de dicho tráfico de información [9].

Estos datos no sorprenden si se tiene en cuenta que detrás de estos grandes números se encuentran gigantes del sector tecnológico como Google, cuya información supera los 10 Exabytes y que cada día sus usuarios realizan más 3.500 millones de búsquedas que serán ejecutadas por uno de sus 1.300.000 servidores, que procesan aproximadamente unos 20 Petabytes al día [10]. Empresas como

Amazon y Facebook disponen de más de 1 Exabyte de información y los 2.700 likes y las 300 millones de fotos subidas generan 0,5 Petabytes a diario en los servidores de la empresa de Palo Alto [11].

Para concluir, hoy en día más del 94% de la información que producimos es en formato digital y alrededor del 90% de la información global jamás generada se ha creado en los últimos dos años [12]. Este último dato nos permite entender la envergadura del fenómeno.

2.1.2 El concepto de Big Data

La palabra Big Data apareció por primera vez en 1997 en un artículo de la NASA [13] en el que se analizaba el problema del creciente incremento del tamaño de los datos y el consecuente problema de visualización de los mismos. Se estaban empezando a dar casos en los que el dataset a visualizar era mucho más grande que la memoria física del sistema, o aún peor, que el propio espacio en disco. Los investigadores de la NASA se refirieron a esta limitación como 'the problem of big data'. En dicho documento se analizaban las técnicas en las que se había investigado para contrarrestar este fenómeno y mejorar el rendimiento del sistema a la hora de visualizar los datos.

El Big Data, por lo tanto, no sólo hace referencia a las grandes cantidades de información que nuestra sociedad produce diariamente, sino también al conjunto de herramientas que se pueden aplicar a sistemas que manejan conjuntos de datos, de gran tamaño o de gran complejidad, para simplificar su análisis o para extrapolar datos sobre los mismos, que de otra forma, no se podrían obtener en tiempos razonables o con la potencia de procesado del que se dispone con los sistemas convencionales.

Un sistema de análisis Big Data no necesariamente implica un conjunto grande de datos. También podría ser un análisis sobre un conjunto de datos bastante limitado, sobre el que hay que aplicar unos algoritmos de alta complejidad que implicarían unos tiempos de ejecución elevados.

El mayor interés que hoy en día hay por las técnicas de análisis de grandes cantidades de datos, evidentemente está muy relacionado con la cantidad de información disponible y gratuita que nuestra sociedad produce.

2.1.3 Origen y tipos de los datos

Con la llegada de los dispositivos móviles el tipo de información que se puede almacenar de cada usuario ha cambiado mucho con respecto a lo que se podía guardar hace unos años. Estos dispositivos incluyen una infinidad de sensores y de coprocesadores que son capaces de registrar todo tipo de datos y añaden profundidad a la información.

Los móviles inteligentes son capaces de guardar la posición a través de GPS, o incluso a través de la conexión 3G o wifi. Pueden tener registro de temperatura, presión, movimiento, orientación, etc. Datos que hasta ahora por sí solos podrían parecer insignificantes, pero a los que se les puede buscar una relación con los patrones de conducta del usuario.

Toda esta información puede ser empleada para mejorar la experiencia de usuario ya que sirven para comprender mejor al cliente y ofrecer un servicio más personalizado y completamente adaptado a las exigencias del mismo. Además, todo estos datos, son muy valiosos porque pueden generar un perfil de conducta del usuario o patrón de compras que puede ser utilizado para estimular las mismas o aumentar el valor medio de la venta.

A su vez, con el avance de la tecnología, se añaden cada vez más sensores y características móviles a otros dispositivos que tradicionalmente no los tenían: televisiones, automóviles, electrodomésticos, domótica, etc. El escenario futuro es que cada uno de los dispositivos de nuestra vida cotidiana tenga su propia conectividad y sus propios sensores, permitiendo guardar datos de todo tipo acerca de la localización, formas de uso y costumbres del usuario.

En el análisis de Big Data, los datos pueden presentar cualquier forma en cuanto a la estructura y el tipo, ya que estos factores no suponen una limitación para el procesado, empleando el framework adecuado.

En general, los datos se pueden dividir en tres categorías:

- **Datos estructurados**, que se guardan de forma ordenada con un índice y con campos que tienen estructura predeterminada y/o valores preestablecidos. Este tipo de datos eran los más comunes hasta la llegada del Big Data y con los que estamos todos acostumbrados a trabajar, ya que son los que se emplean en las bases de datos relacionales, tipo MySQL o en las tablas.

- **Datos no estructurados,** que se almacenan en el mismo estado con el que se capturaron y que carecen de estructura preestablecida. En esta categoría podemos encontrar: emails, mensajes de texto, redacciones, imágenes, enlaces, documentos multimedia, PDF, etc. Es una categoría con la que estamos muy acostumbrados a trabajar, pero a partir del desarrollo de las técnicas de Big Data ha cobrado mucha más importancia ya que ahora somos capaces de gestionar y analizar en reducido tiempo un gran volumen de este tipo de datos.
- **Datos semiestructurados**: aquellos datos que presentan parte de su cuerpo de forma estructurada, por lo que tienen separación de elementos que a su vez pueden ser datos estructurados o no estructurados y que permiten una mayor flexibilidad. Esta categoría es el paradigma de los nuevos tipos de datos, ya que representan una forma muy eficaz de relacionar elementos y datos no estructurados a la vez, siendo un ejemplo muy válido los documentos en formato JSON, XML, HTML, etc.

El análisis de Big Data no se limita a una categoría en concreto, ya que puede hacer uso de distintos tipos de datos, fusionando conceptos y adoptando algunas soluciones para determinadas tareas y otras para otros problemas. Sin embargo, el tipo de datos semiestructurado es el que más se emplea debido a su alta flexibilidad, ya que permite crear esquemas y relaciones sin tener que estructurar todos los datos.

2.1.4 Uso de Big Data

Hasta ahora se ha intentado explicar el porqué del uso de estas herramientas sin detallar con qué fin se usan estas técnicas para analizar los datos. El uso que se puede dar a estos sistemas es inimaginable y se aplica en muchos sectores, desde las finanzas y la economía, a la medicina y la astronomía. En esta sección se intentará esbozar brevemente las principales aplicaciones del análisis de Big Data.

El sector empresarial es, sin duda alguna, el que hace un mayor uso del Big Data, ya que permite, entre otras cosas, analizar el perfil de clientes y de su conducta.
Muchas empresas optan por el uso de sus propios datos para crear perfiles de clientes que pueden ser integrados con fuentes externas gratuitas como redes sociales o proporcionadas por terceros. El objetivo final es crear modelos predictivos que permitan adelantarse a las necesidades del cliente, vender productos justo en el momento en el que se produce la necesidad (adelantándose a la competencia y aprovechando un mayor coste de oportunidad, obteniendo

mayor ganancia) y por último, pero no por eso menos importante, dirigiendo al cliente publicidad ad hoc que coincide con el perfil generado, para que el targetting sea más preciso y sobre todo más eficaz.

También en el mundo empresarial, el Big Data se emplea para entender y optimizar los procesos de negocio. Con estas herramientas se pueden generar modelos de consumo que estiman las predicciones de venta a partir de otras variables como las tendencias de mercado o los cambios climáticos, optimizando la cadena de suministro y por lo tanto mejorando la logística y la distribución. Al añadir sensores de radio frecuencia o sensores de posicionamiento GPS, se pueden trazar mapas de distribución que posteriormente pueden ser analizados y mejorados, optimizando costes, evitando desperdicios y redundancia innecesaria.

En el campo médico las ventajas de Big Data son muchas y muy importantes. Toda la medicina genética se ha visto altamente beneficiada por la aparición de software capaz de tratar un gran volumen de información que permite secuenciar con más facilidad grandes cadenas de ADN en muy poco tiempo. Los estudios que se están realizando en este campo permitirán conocer mejor los patrones de comportamiento de las enfermedades y su relación con el genoma, pudiendo encontrar curas a dichos problemas. Además, añadiendo sensores 'wearables' que monitorizarán continuadamente nuestras constantes vitales podremos cruzar miles de datos y adelantarnos a posibles problemas médicos futuros, haciendo la prevención más eficaz y con menor necesidad de supervisión de personal especializado.

Parece ciencia ficción, pero en algunos hospitales se está empezando a monitorizar población sensible, como los recién nacidos, obteniendo un mapa de relaciones entre constantes vitales y síntomas con enfermedades. De esta forma se han creado algoritmos que son capaces de adelantarse a la aparición de los síntomas más graves o de las afecciones mismas.

Otro sector en el que se aplica el Big Data es el mundo deportivo. A través de estos algoritmos se pretende analizar las prestaciones de los jugadores y de los equipos, de tal forma que se puedan detectar fallos en los entrenamientos y en los partidos y poder corregirlos a tiempo, evitando lesiones y/o derrotas.

El campo de la ciencia, que ha sido el que ha visto nacer este tipo de análisis de datos, también se está beneficiando de las prestaciones y características del Big Data. Los experimentos científicos se hacen cada vez más complejos y pretenden relacionar más variables, generando una cantidad de datos que crece de forma

exponencial. Para poner un ejemplo concreto, el CERN, con sus 65.000 procesadores, genera unos 30 Petabytes de información que necesitan de un framework específico para gestionar dicha cantidad de información.

El Big Data se usa también en el campo de la seguridad, sobre todo a nivel de inteligencia nacional, para poder detectar posibles amenazas terroristas o prevenir ataques informáticos. Siempre en este mismo ámbito las compañías aseguradoras y de crédito emplean técnicas de Big Data para detectar casos de uso fraudulento de tarjetas o de seguros.

Parece una paradoja, pero los mismos centros de datos donde se procesan la mayoría de Petabytes de información que producimos, generan grandes cantidades de datos: consumos, peticiones de otros datos, tiempos de ejecución, tasas de aciertos y de fallos al ejecutar procesos u obtener recursos. Toda esta información también es monitorizada y analizada por algoritmos capaces de optimizar el uso de recursos, con el principal objetivo de ahorrar energía y mejorar los servicios prestados.

Las principales metrópolis del mundo crecen cada vez más, complicando aún más el control y la gestión de los recursos disponibles en ellas. El tráfico es uno de los principales problemas y con el análisis de Big Data se pueden hacer modelos de tráfico a partir de observaciones reales. Esto permite a las entidades municipales planificar la construcción de grandes infraestructuras para paliar dicho problema. Además, de una forma más simple, se puede monitorizar el uso del transporte público pudiendo crear un modelo de flujo de pasajeros y reforzar determinadas líneas de metro o autobús, que en días puntuales o en franjas horarias predeterminadas, presentan un pico de afluencia mejorando así el servicio.

Por último, pero no por ello menos importante, el análisis de Big Data se emplea en el sector financiero, sobre todo en la negociación de alta frecuencia donde se emplean algoritmos muy sofisticados de análisis de valores. Estas herramientas permiten tomar decisiones sobre la compra o venta de títulos de una cartera real, pudiendo tomar una decisión y realizar la transacción en menos de un microsegundo.

Como se puede ver, por lo que se ha comentado en esta sección, los usos que se pueden dar a estas aplicaciones son múltiples y muy variados desde el campo científico/sanitario hasta el económico/financiero. Además, como la tecnología está siempre en evolución, el número de aplicaciones que surgen se multiplican.

En resumen, siempre que haya un sistema que genere muchos datos se pueden aplicar técnicas de Big Data para obtener conclusiones que permitan optimizar procesos y recursos.

2.1.5 Business Intelligence y diferencias con el Big Data

El concepto de Business Intelligence (BI) es bastante más antiguo que el de Big Data, ya que se creó a mediados del siglo XIX y se refiere al análisis de datos, de más o menos complejidad, empleando estadística descriptiva. Esto permite medir indicadores estadísticos tales como la media, la varianza, la desviación o la moda. Los dataset suelen estar caracterizados por datos limitados, filtrados y que siguen modelos simples.

La principal diferencia con el Big Data reside en que éste último hace uso de estadística inferencial y los datos sobre el que se aplica el análisis son muy heterogéneos y no están necesariamente correlacionados entre ellos. El objetivo principal del Big Data, por lo tanto, es encontrar aquellas relaciones ocultas, relaciones no lineales, dependencias causales, e inclusive casuales, que pueden existir entre las distintas variables aleatorias que se tienen en consideración para el análisis.

Esta diferencia conceptual no implica que hoy en día, a la hora de hacer análisis de Business Intelligence, se puedan aplicar técnicas de Big Data para profundizar en los datos o para fusionar los datos de BI con otros, de diferentes ámbitos.

2.2 Topic Modeling

2.2.1 Introducción

Conforme nuestra sociedad va produciendo más cantidad de información, se ha visto que no tenemos la capacidad humana para categorizar todos los documentos de los que disponemos. Para ello, los investigadores de la rama de Aprendizaje Automático (Machine Learning) han desarrollado una serie de algoritmos capaces de trabajar con grandes colecciones de documentos y asociar a cada uno de ellos un determinado tema. Este tipo de análisis recibe el nombre de Modelado Probabilístico de Temas (Probabilistic Topic Modeling).

Dichos algoritmos son métodos estadísticos que analizan las palabras de los textos para encontrar los temas de los que se componen dichos documentos, cómo se relacionan entre ellos y cómo cambian a lo largo del tiempo. Una de las principales ventajas de estos algoritmos es que no necesitan un etiquetado previo de los documentos y los tópicos se encuentran directamente de forma automática en el texto.

De allí que el Topic Modeling consiste en identificar patrones en un corpus, en concreto es un método para buscar e identificar los grupos de palabras que constituyen los temas en grandes colecciones de textos. Estos temas se pueden definir como patrones recurrentes de palabras concurrentes y dichos algoritmos de minería de textos se aprovechan de las relaciones ocultas entre las palabras.

2.2.2 Historia

Uno de los primeros pasos hacia la generalización del Topic Modeling, se desarrolló para calcular la frecuencia de aparición de las palabras dentro de un texto. Esta tarea parece fácil, pero en seguida los investigadores se encontraron con el problema de que al asociar a cada palabra un número de apariciones en el texto y al ordenarlas por cantidad de repeticiones, las primeras palabras eran del tipo: "de", "la", "que", "el", "y", "a", "los", "se", "del" (en este ejemplo las 10 palabras más usadas en castellano). Estos términos toman el nombre, en inglés, de stopwords y son palabras que no aportan significado léxico por lo que se suelen eliminar o ignorar a la hora de analizar textos.

Para poder medir de forma más certera la frecuencia de aparición de las palabras relevantes, se creó el concepto de frecuencia de término/frecuencia inversa de documento (*Term frequency/Inverse document frequency* Tf-idf) que permite

ponderar las apariciones de una palabra por la frecuencia de aparición en el corpus. Este algoritmo y sus variantes son muy usados en el campo de la minería de datos (Data Mining) y en los motores de búsqueda, para poder ponderar a la hora de realizar búsquedas dentro de grandes colecciones de documentos.

El algoritmo de Topic Modeling fue muy importante para todo el desarrollo posterior ya que es el punto de partida para los trabajos siguientes. En concreto, en 1990 Scott Deerwester et al. publicaron Indexing by Latent Semantic Analysis (LSA) [14], en el que se postulaba un sistema que empleaba los resultados del algoritmo Tf-Idf en una matriz de palabras de documentos muy grandes. Cada fila es una palabra del diccionario, cada columna representa un documento y el resultado del algoritmo es la intersección de cada documento y palabra. Este proceso, que parece bastante trivial e intuitivo, básicamente es el esquema que se sigue empleando hoy en día. A partir de esta matriz, también llamada en la literatura como 'bag of words', LSA aplica el principio algebraico de descomposición en valores singulares (Singular Value Decomposition SVD) que consiste en la factorización de la matriz, para obtener la relación entre cada una de las palabras y, sobre todo, para conocer cuáles son las palabras que se usan más a menudo con otras en un mismo documento.

Habrá que esperar hasta 1998 para la aparición de un primer trabajo sobre Topic Modeling: Latent Semantic Indexing, Christos H Papadimitriou et al. [15]. Este primer trabajo de carácter teórico, ponía el foco de atención en el análisis espectral de la matriz de términos de un documento para poder obtener información.

Hofmann, a partir del LSA, desarrolló un nuevo algoritmo que dejaba de lado el enfoque del álgebra lineal de éste último y se centraba en el análisis estadístico y de los modelos probabilísticos publicando en 1999 Probabilistic Latent Semantic Analysis (pLSA) [16]. La gran novedad que se introdujo con este nuevo estudio fue la creación de una nueva relación entre las palabras y los documentos: el tema. Las palabras se distribuyen alrededor de temas que pueden tener más o menos palabras y pueden compartir palabras con otros temas, dentro de cada documento. Además, puede haber uno o más temas, cada uno con una relevancia diferente y específica del texto.

Como se puede intuir, los temas no son una entidad preestablecida y tampoco una variable tangible y medible ya que se deben deducir directamente a partir del texto, así como un ser humano lo haría al leer un documento. Además, este novedoso método de aprendizaje no supervisado hace uso del algoritmo iterativo

de Esperanza-Maximización (Expectation Maximization Algorithm EM) para determinar la distribución de las variables latentes.

En 2003 Blei et al. [17] retomaron esta idea de la estructura latente de temas y diseñaron un modelo generativo de documentos llamado Latent Dirichlet Allocation (LDA). Este nuevo método sentó un precedente en el campo de Topic Modeling ya que muchísimos trabajos posteriores lo tomaron como punto de partida. A día de hoy sigue siendo el algoritmo más usado, por lo que en esta memoria se le dedica una sección entera (véase 2.2.3) en la que se detalla su funcionamiento con más detenimiento.

En 2004 Michal Rosen-Zvi et al. [18] publicaron el primer trabajo en el que se planteaba la inclusión de metadatos al algoritmo LDA. En concreto, en este método se incluye en el modelo la información del autor, relacionando por lo tanto al autor con los temas, a través de una distribución multinomial o una combinación de distribuciones en el caso de múltiples autores. En este trabajo se tomó como corpus 1.600 publicaciones científicas completas y 160.000 resúmenes de trabajos científicos en los que la relación entre autores y temas es bastante significativa.

En 2005 McCallum et al. [19] plantearon una extensión de este último algoritmo para adaptarlo a corpus procedentes de emails, cartas y posteriormente en McCallum et al., 2007 [20] a redes sociales. En este sistema Autor/Receptor-Tópico (Author-Recipient-Topic ART) se establece una nueva relación ya que, además del autor, se puede fácilmente identificar también el posible destinatario de la comunicación y los temas se infieren a partir del conocimiento del trasmisor y receptor del mensaje.

Otros ejemplos de inclusión de metadatos en modelos generativos LDA son el trabajo de Dietz et al., 2007 [21] que estudia la relación entre documentos y las citas que aparecen en ellos y el trabajo de Nallapati et al., 2008 [22] que, además de crear un modelo de similitud de temas entre documentos citados y documentos que citan a otros, es capaz de predecir posibles citas que no están presentes en los mismos.

2.2.3 Latent Dirichlet Allocation

2.2.3.1 Introducción

Este algoritmo es uno de los más simples y más usado en Topic Modeling y se basa en el concepto básico de que una colección de documentos se divide en un número determinado de temas y que cada documento de dicha colección

presenta per se una estructura lógica de palabras que pertenecen, en diferente proporción, a cada uno de los temas.

Como es evidente, las palabras que componen los documentos son observables, mientras que los temas en los que se divide el corpus, la relación entre cada documento y el tema, el nexo entre cada palabra de las colecciones y el tema, son estructuras no explícitas. Por lo tanto, el reto computacional que establece esta familia de algoritmos es desvelar la estructura latente de temas que hay detrás de cada corpus, a partir de los documentos observables que lo componen.

LDA, como otros algoritmos de Topic Modeling, pertenece al campo de Probabilistic Modeling, que necesita de un modelo generador que sea capaz de crear valores aleatorios a partir de un dato observable, teniendo algunos parámetros ocultos. Este proceso generativo permite modelar una función de densidad de probabilidad conjunta que incluye tanto los datos observados como los parámetros ocultos y que como objetivo último permite obtener la distribución condicional de las variables latentes, siendo conocidas las variables observadas. Esta distribución condicional, en estadística Bayesiana, es también conocida como probabilidad a posteriori (Posterior Distribution). Como el nombre indica es la probabilidad condicionada que se asigna después de que un hecho observable se ha tomado en cuenta, como una variable no latente.

Aplicando estos conceptos a la tarea de análisis de texto en busca de los temas que lo componen, las variables observables serían las palabras que conforman el corpus, las variables latentes vendrían dadas por la estructura de los temas, incluyendo las relaciones entre palabras, documentos y temas, mientras que el proceso generativo sería el cómputo de la distribución condicional de las variables latentes a partir de los textos. Computacionalmente el algoritmo se reduce al cálculo de la probabilidad a posteriori.

2.2.3.2 Proceso generador

Para un análisis más detallado del proceso generador y del funcionamiento del algoritmo LDA es necesario describir la notación que se empleará y definir algunos conceptos.

La palabra se define como la unidad básica de información discreta que será parte de un diccionario, por lo que presentará un índice dentro del mismo.

Si consideramos un vocabulario como un vector podremos denotarlo como $V = \{w_1, w_2, \dots, w_V\}$ donde w_i son las palabras que lo constituyen y el subíndice denota el registro que ocupa dentro del mismo.
El orden de las palabras dentro de un diccionario no es relevante para el algoritmo y el subíndice v denota el tamaño en palabras del diccionario.

Definimos un documento como un conjunto de palabras $D = \{w_1, w_2, \dots, w_N\}$ donde w_n corresponde a la n-ésima palabra dentro de la secuencia del texto. N denota por lo tanto el número de palabras presentes en el documento. Un corpus se considera como la colección de documentos que engloba todos los textos sobre los que se quiere realizar el análisis y se denotará como $C = \{d_1, d_2, \dots, d_M\}$ dónde d_m es el m-ésimo documento dentro del corpus y M es el número de documentos presentes en el corpus.

Si definimos los temas como un vector $\beta_{1:K}$ consideramos cada β_k como una distribución sobre el vocabulario V siendo K el número de temas o tópicos presentes en el corpus. A la presencia del tópico $k\text{-}ésimo$ en un determinado documento d la llamaremos $\theta_{d:k}$. Denotamos las asignaciones de una palabra n de un documento d a un determinado tema como $z_{d,n}$ y las palabras que observamos en un documento d se denotarán como w_d denotando la palabra $n\text{-}ésima$ de dicho documento como $w_{d,n}$.

Teniendo en cuenta las anteriores notaciones, el proceso generador para el algoritmo LDA corresponde a la siguiente distribución de probabilidad conjunta de las variables observables y latentes:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{k=1}^{K} p(\beta_k) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right)$$

Como se puede fácilmente observar dicha distribución conjunta presenta dos dependencias: por un lado la asignación de la n-ésima palabra del documento d $z_{d,n}$ a un determinado tema, depende de la proporción de cada tema en cada documento θ_d y, por otro lado, la palabra observada $w_{d,n}$ depende tanto de la asignación de tópicos $z_{d,n}$ como de todos los tópicos $\beta_{1:K}$.

2.2.3.3 Representación del modelo gráfico probabilístico

Para poder apreciar mejor la relación entre las distintas variables, la figura 2.1 representa un modelo gráfico probabilístico (Probabilistic Graphical Model PGM)

donde se muestran las dependencias condicionales de las distintas variables aleatorias que intervienen en el proceso generativo del algoritmo LDA.

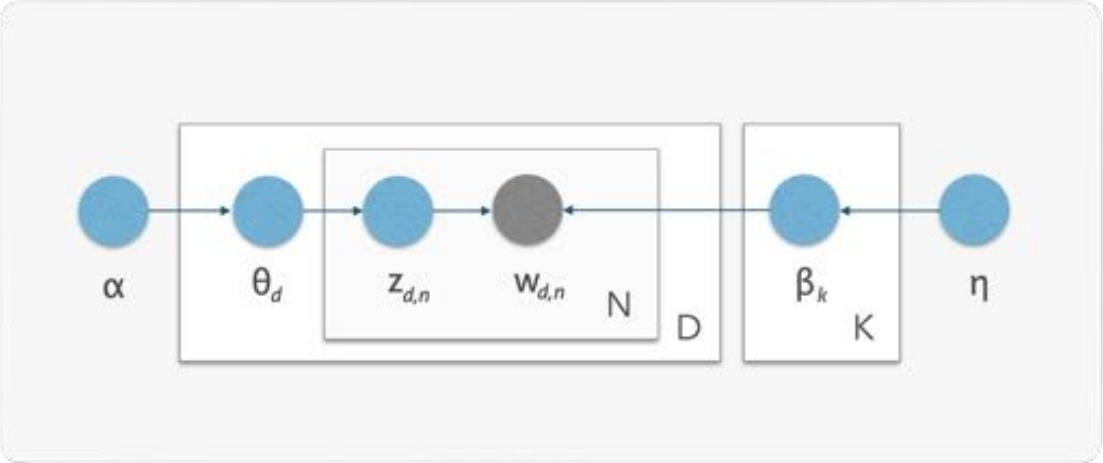


Figura 2.1: Modelo Gráfico Probabilístico.

Cada nodo representa una variable aleatoria, los nodos blancos representan variables latentes como la proporción de cada tema en un documento, la asignación de cada palabra del documento a un tema y los propios temas. El nodo sombreado representa la variable no oculta y observable: las palabras del documento. Esta forma de representar un proceso aleatorio también recibe el nombre de Plate Notation y cada rectángulo o lámina (Plate) representa las repeticiones de las variables presentes en el mismo. Por ejemplo, la lámina N representa las repeticiones que se verifican por cada palabra presente en el documento, mientras que la lámina D representa las repeticiones dentro del corpus. Obviamente la lámina K representa las K repeticiones que generan todos los temas del vector β.

Los nodos que están fuera de las láminas representan las variables que no sufren repetición y que de alguna forma configuran las entradas del algoritmo y, en concreto, el parámetro α representa un factor de concentración de la distribución Dirichlet a priori $Dir(\alpha)$ de la distribución de los temas por cada documento. Este parámetro de concentración se define de la siguiente forma:

$$\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_K\}, \qquad \alpha_i \in \mathbb{R}^+, \qquad K > 2$$

y se podría considerar como el peso a priori de cada tema k en un documento, normalmente igual para todos los K tópicos y de valor bastante pequeño para favorecer una asignación de pocos temas a cada documento, haciendo éstos menos dispersos y disminuyendo el error.

El parámetro η (también citado en la literatura como β, y que no se debe confundir con el vector de temas) es el parámetro de la distribución a priori Dirichlet $Dir(\eta)$ de la distribución de los temas por cada palabra.

$$\eta = \{\eta_1, \eta_2, \dots, \eta_V\}, \qquad \eta_i \in \mathbb{R}^+, \qquad V > 2$$

y podría considerarse como el peso a priori que se le puede asignar a cada palabra en un tema, normalmente igual para todas las palabras y también éste de valor bastante bajo, para favorecer que cada tema presente pocas palabras, haciendo los temas más consistentes y mejor definidos.

2.2.3.4 Cálculo de la probabilidad condicional

Como se ha mencionado anteriormente, lo que se necesita es poder calcular la probabilidad condicional de la estructura de temas sobre las variables observables, incluyendo las variables latentes. Dicha probabilidad a posteriori (también llamada $posterior$ en la literatura) se calcula como la probabilidad condicionada de las variables latentes conociendo la variable no oculta $w_{1:D}$.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

El numerador es la distribución conjunta de todas las variables aleatorias involucradas en el proceso, que puede ser fácilmente calculada para cualquier configuración de las variables latentes. El denominador es la probabilidad marginal (también llamada $evidence$ en la literatura) y representa la probabilidad de observar la parte no latente del corpus en cualquier modelo de temas. A nivel teórico es fácil de calcular ya que corresponde a la suma de todas las posibles distribuciones conjuntas generadas por cada instancia de la estructura latente.

El problema de este cálculo es que, con la tecnología actual, computacionalmente es imposible sumar todas las posibles distribuciones, ya que el número de estructuras distintas es desmesurado. Por lo tanto, podemos afirmar que el mayor esfuerzo se ha hecho, y se sigue haciendo, para mejorar las aproximaciones de la probabilidad marginal, evitando así el cálculo teórico de dicho valor. Hay dos tipos de algoritmos que se emplean para aproximar la probabilidad marginal: algoritmos basados en el muestreo o basados en métodos variacionales.

Los métodos de muestreo (Sampling-Based Algorithms) intentan obtener el mayor número de valores de la distribución a posteriori para poder aproximarla a una distribución empírica. El algoritmo más usado de esta familia en Topic Modeling

es el algoritmo de muestreo de Gibbs que genera una cadena de Markov de muestras correladas con los valores obtenidos y aproxima la distribución marginal de las variables latentes.

Los algoritmos de métodos variacionales, a diferencia de los de muestreo, son de tipo determinista y no aproximan la distribución a posteriori con muestras. Se caracterizan por postular una familia de distribuciones parametrizadas sobre la estructura latente y buscar cuál de todas estas distribuciones es la que más se parece a la distribución de probabilidad a posteriori. La ventaja de este tipo de algoritmos es que transforman un problema de muestreo en una tarea de optimización, consiguiendo una mejora de las familias parametrizadas de distribuciones, mejorando los modelos y, a su vez, permitiendo que el algoritmo LDA sea más eficaz a la hora de individuar los temas. Algunos ejemplos de estos algoritmos son Variational Bayesian Inference (VB), Expectation Propagation (EP), Variational EM, entre otros.

2.2.3.5 Otras implementaciones de LDA

Como se ha visto en la introducción de este capítulo, a lo largo de los últimos años se han generado muchos estudios en los que se ha añadido más profundidad al análisis de LDA, incluyendo metadatos al corpus de documentos. Pero hasta ahora el esquema de fondo siempre era la relación entre palabras, documentos, otras variables no latentes y la estructura oculta de temas. El concepto se puede abstraer un poco más y, con algunos cambios en los algoritmos a emplear, se obtiene un modelo heterogéneo de datos agrupados, donde en lugar de asociar cada grupo observado con un elemento, se asocia cada grupo a múltiples elementos con diferentes pesos. Este nuevo planteamiento permite aplicar el concepto que hay detrás de LDA a otros tipos de datos como audio, música, logs, información de redes sociales, ADN, o código de programación, entre otros.

Algoritmos basados en este enfoque se emplean habitualmente en Visión Artificial (Computer Vision), donde se aplica el algoritmo EM Variacional que se adapta para hacer búsqueda de imágenes, clasificación y organización. Asumiendo que una imagen es una colección de patrones visuales y que cada una de ellas contiene una diferente proporción de dichos patrones, es posible clasificarlas [23], relacionar imágenes y pies de imagen [24] y crear estructuras jerárquicas de imagen [25][26] para clasificar, organizar o reconocer objetos.

Otro campo donde se ha empleado con éxito algoritmos tipo LDA es en genética de poblaciones, para poder encontrar el origen de las poblaciones ancestrales. La

idea subyacente es que el genotipo de cada individuo puede pertenecer a una o a más poblaciones ancestrales, por lo que cada persona podría ser considerada como una palabra en un documento y cada población ancestral estaría caracterizada por unos patrones genéticos que serían los temas, siguiendo con la analogía de la estructura clásica de Topic Modeling, que a su vez relacionan cada individuo con dichas poblaciones en menor o mayor medida. Este enfoque resulta muy útil ya que los patrones genéticos que se supone pertenecen a una población ancestral no son conocidos y serían una de las variables latentes que los algoritmos de tipo LDA intentan relacionar con las variables observadas.

Otros enfoques de LDA se pueden obtener liberando el modelo de algunas suposiciones iniciales que limitan la aplicabilidad de este algoritmo a estructuras de texto más complejas. Una de estas consideraciones previas es que el orden de las palabras no importe y se tome una 'bolsa de palabras' (Bag of Words) de un diccionario que están ordenadas de una forma distinta a como aparecen en el texto. Estas consideraciones son muy importantes, por ejemplo, si se pretende analizar la estructura semántica del texto. Con dicho fin se han propuesto unas modificaciones al algoritmo LDA que toman en consideración el orden de las palabras, condicionando la generación de temas a los vocablos que preceden las palabras, como se puede observar en el trabajo de Wallach [27].

Otra consideración previa es que el orden de los documentos dentro del corpus tampoco afecta al algoritmo, pero para poder analizar la evolución de un tema a lo largo de los años es necesario tener una referencia cronológica de los documentos y por lo tanto tienen que estar ordenados por fecha de publicación. Dynamic topic modeling [28] de David Blei explora esta posibilidad, dejando claro que no sólo se puede considerar un tema como una distribución a lo largo de un conjunto de palabras, sino que se puede ver como una secuencia de distribuciones que van cambiando con el tiempo.

Por último, en el algoritmo LDA, se considera como suposición que el número de temas es conocido y fijo. En modelos de tópicos Bayesianos no-paramétricos se propone definir el número de temas mientras se calcula la probabilidad a posteriori. Este supuesto permite añadir nuevos temas conforme vayan apareciendo nuevos documentos. Otros estudios [29] han analizado también cómo incluir el concepto de Jerarquía de tópicos en los modelos Bayesianos no-paramétricos. Este enfoque permite crear un árbol de temas que van desde los más genéricos a los más concretos, estructura que se infiere directamente desde los datos.

Hay otras extensiones de LDA que tienen en cuenta la correlación de palabras entre temas y también modelos que introducen la posibilidad de que una palabra no pertenezca a determinados temas.

2.2.3.6 Conclusiones

LDA es un algoritmo muy usado en el campo de Topic Modeling debido a su eficacia y que, a nivel conceptual, se basa sobre una idea intuitiva y bastante genérica como es el concepto de estructura latente de temas dentro de una colección de documentos. Dicha 'simplicidad' ha permitido crear extensiones del algoritmo aplicables a otros ámbitos de Topic Modeling e incluso ampliar su aplicabilidad a otros campos de la ciencia de la computación.

2.3 Framework

2.3.1 El problema computacional

Como se ha visto en la sección anterior, el algoritmo LDA necesita calcular la distribución a posteriori y la obtiene a través de muestreo de valores o de inferencia mediante un algoritmo variacional. La complejidad para encontrar la estimación del máximo a posteriori en LDA está estrechamente relacionada con el número de temas que hay en cada documento (no confundir con el número total de temas presentes en la colección de documentos). En concreto, dicha complejidad es de clase P, de orden polinomial $T(n) = O(n^k)$ siempre que el número de temas por documentos no sea muy grande. En cuanto se aumenta el número de temas presente en cada texto, la complejidad aumenta y pasa a ser de clase NP-hard. Este problema computacional se suma al tamaño de la entrada, ya que dependiendo del tamaño del corpus, la matriz de frecuencias que se deberá analizar será mayor, siendo sus filas las V palabras del diccionario y las columnas los D documentos presentes en la colección.

El algoritmo LDA se ha desarrollado en múltiples plataformas usando diferentes lenguajes de programación desde implementaciones en C [30], C++, Python, R y Java [1], entre otros.

Con la aparición de frameworks específicos para poder analizar grandes conjuntos de datos, se han hecho experimentos en este sentido y se han creado implementaciones para plataformas que permiten ejecutar aplicaciones distribuidas. El framework más popular para crear aplicaciones distribuidas de análisis de Big Data es sin duda alguna Hadoop [2], del que se detallan en la siguiente sección (véase 2.3.2) sus características y funcionamiento.

2.3.2 Apache Hadoop

2.3.2.1 Introducción

Hadoop es un proyecto de software libre de Apache Foundation, que fue creado para poder realizar cálculos extensivos sobre ficheros de gran tamaño o grandes colecciones de ficheros en un entorno clusterizado. Uno de los objetivos principales de esta herramienta es poder ejecutar algoritmos paralelizables que puedan correr en distintas máquinas y asegurar que la aplicación resultante sea completamente tolerante a fallos. Este último requisito se debe a que la computación distribuida aumenta la posibilidad de que uno de los nodos pueda

generar errores de ejecución o de conexión, puesto que no necesariamente están físicamente cerca. Este proyecto fue inicialmente creado por Doug Cutting y está inspirado en el modelo de programación MapReduce [10] introducido por Google así como en The Google File System (GFS) [31], que se describen en el siguiente apartado (véase 2.3.2.2).

Dadas sus múltiples ventajas son muchas las empresas que hacen uso de forma extensiva de este framework, entre las que destaca sin duda alguna la contribución de Yahoo!, que ha aportado al proyecto recursos y prestigio al integrarlo en 2008 en los buscadores de la compañía [32], demostrando la gran capacidad de este sistema para gestionar un billón de enlaces y 5 Petabytes de información con 10.000 núcleos dedicados. Actualmente Yahoo! utiliza más de 100.000 núcleos para los servicios de publicidad y búsqueda. Por su parte, Facebook también emplea unos 11.000 núcleos en 2 clústeres con Hadoop para guardar los logs y poder hacer análisis y generar reports; eBay, Linkedin y Amazon [33] son sólo algunos ejemplos de la consolidación de esta herramienta en el mercado.

2.3.2.2 GoogleFS y MapReduce

Google File System o GoogleFS nació de la necesidad de actualizar el anterior sistema de archivos (BigFiles) que empleaba el coloso de Silicon Valley, ya que la cantidad de información que Google generaba ya por aquella época hacía necesario mejoras en el software, pero sin tener que abordar cambios drásticos en el hardware.

En este trabajo [31] se propone un sistema distribuido y escalable de gestión de archivos para aplicaciones que hacen uso extensivo de datos. Una de las ventajas de este sistema es que permite ejecutar aplicaciones tolerantes a fallos y que, además, no requieren de un sistema hardware específico, permitiendo accesos múltiples simultáneos. Los archivos se dividen en trozos muy grandes en tamaño (64 Megabytes habitualmente), mientras que el clúster se divide en múltiples nodos que pueden ser de tipo Worker (conocido también como Chunkserver), que son los que se encargan de almacenar los fragmentos de tamaño fijo, normalmente con un factor de redundancia, o de modo Maestro (Master) que guarda la identificación de los trozos y se encarga de recibir las peticiones externas de las aplicaciones a los datos. Una de las principales características de este sistema de gestión de ficheros es que no está implementado a nivel de kernel de sistema operativo y funciona como una librería externa.

MapReduce [10] es un paradigma de programación, diseñado específicamente para computación distribuida, donde hay un gran número de servidores y se basa en dos macros Map y Reduce, que permiten paralelizar con más facilidad algoritmos para que puedan ser ejecutados en un clúster de servidores. A continuación se describe el funcionamiento de este paradigma de programación.

El sistema de archivos o el framework sobre el que se lanza este tipo de función divide el fichero de entrada en trozos de 64 Megabytes, normalmente almacenados en una estructura del tipo $< clave, valor >$, y se encarga de lanzar una instancia de MapReduce en cada uno de los nodos, incluyendo el maestro. Este último es el que se encargará de gestionar el flujo de datos hacia cada Worker y la asignación de tareas, ya que pueden estar realizando tanto el Map como el Reduce en cualquier momento. En caso de fallo, el Master será el que se encarga de parar el flujo de un Worker y asignarle otro flujo, haciendo la aplicación completamente tolerante a fallos.

Los nodos encargados de realizar tareas de Map generan tuplas $< clave, valor >$ con el resultado de la función a realizar, a partir de la porción de datos asignada y la guardan en memoria. Dichos resultados temporales se guardan en disco periódicamente y el master es el que se encarga de asignar cada uno de estos resultados parciales a los distintos Workers que estén realizando tareas de Reduce. Estos últimos se encargan de agrupar todos los datos que comparten la misma clave para aplicar una función determinada sobre ese conjunto de datos y añaden una nueva pareja $< clave, valor >$ al fichero de salida. Cuando todos los nodos de Map y Reduce han terminado, el maestro puede juntar todos los ficheros de resultados, normalmente uno por cada instancia Reduce que se haya producido en el sistema, en una única salida.

2.3.2.3 Arquitectura de Hadoop

Hadoop es un framework de procesado distribuido que hace uso de almacenamiento distribuido y se divide en 4 módulos principales:

- **Hadoop Common:** incluye las principales rutinas para el soporte de los demás módulos de Hadoop.
- **Hadoop Distributed File System**: es el sistema de gestión de ficheros distribuido que emplea para almacenar la información.
- **Hadoop YARN:** es la herramienta que se encarga de la gestión de procesos, tareas y de recursos del clúster.
- **Hadoop MapReduce:** es el sistema que se encarga del procesado distribuido.

A continuación se describirán los módulos más significativos para el análisis de Big Data, que son el sistema de archivos y el sistema de procesado distribuido.

Hadoop Distributed File System (HDFS) fue desarrollado tomando como punto de partida el proyecto GoogleFS (descrito en el párrafo 2.3.2.2), con algunos cambios para que pudiera ser más tolerante a fallos. Este sistema de archivos es distribuido, escalable y está programado en Java. Se compone de un nodo maestro llamado Namenode, con redundancia debido a la extrema sensibilidad de la información que opera, y un clúster de nodos esclavos llamados en esta arquitectura Datanodes, que también tienen cierto grado de redundancia aunque no sea tan crítica como en el caso del nodo maestro. En concreto, el sistema de ficheros prevé que pueda haber hasta 3 copias de la misma sección de datos, de las cuales 2 se mantienen dentro de nodos en un mismo rack y una tercera en un rack externo.

El nodo maestro se comunica con los Datanodes mediante un protocolo específico de envío de bloques de datos utilizando TCP/IP para la comunicación entre ambos.

Para los nodos maestros puede utilizarse commodity hardware que incluye el sistema operativo GNU/Linux y el software específico del Namenode, que otorga la capacidad a este nodo de:

- Gestionar el espacio de nombres del sistema de ficheros.
- Dirigir el acceso de los nodos esclavos a los datos.
- Ejecutar operaciones sobre el sistema de ficheros tales como la creación, renombrado, borrado de datos y directorios.

Para los nodos esclavos también se utiliza commodity hardware e incluyen un sistema operativo GNU/Linux con software específico para realizar tareas de:

- Lectura y escritura sobre los datos.
- Generación de bloques, borrado y réplica siguiendo las instrucciones del Namenode.

Una de las ventajas del framework Hadoop es su completa flexibilidad con el sistema distribuido de ficheros, por lo que aunque se haya diseñado un sistema específico como HFDS para su uso conjunto con la herramienta, éste permite obtener los datos desde otros clusters distribuidos de almacenamiento, que pueden estar en los siguientes formatos: Amazon S3, CloudStore, Windows Azure Storage Blobs (WASB), usando conexión FTP a otros servidores e inclusive a través de HTTP.

Hadoop MapReduce es el módulo que se encarga del procesado distribuido dentro del framework y está constituido por un rastreador del trabajo (Job Tracker) y por varios rastreadores de tareas (Task Tracker) por cada nodo del clúster. Uno de los objetivos principales es conseguir que cada Task Tracker pueda trabajar sobre datos que estén ubicados en la misma máquina, para evitar saturar la red con un excesivo tráfico y limitar el tiempo de carga de los datos procedentes de otra máquina. Dicha gestión de asignación de tareas es llevada a cabo por el Job Tracker que se encarga de asignar tareas que impliquen datos localmente cercanos y reasignar tareas en caso de fallo.

2.3.2.4 Otros proyectos

Debido a que Hadoop es un framework muy popular para la computación distribuida en el campo de Big Data y pertenece a la fundación Apache, ha sido completado con muchos otros proyectos relacionados que constituyen un '*ecosistema de aplicaciones'* para el análisis de Big Data.

En cuanto a los proyectos integrables con Hadoop habría que mencionar:

- **Cassandra:** una base de datos no relacional (NoSQL) distribuida y altamente escalable que permite guardar datos en el formato $< clave, valor >$, y que divide la información en nodos que se comunican mediante el sistema P2P, sin necesitar de un nodo maestro.
- **HBase:** una base de datos distribuida que soporta datos estructurados de tablas muy largas.
- **Mahout:** una librería de Data Mining y Machine Learning para Hadoop.
- **Pig:** una plataforma de alto nivel para crear programas paralelizables empleando el paradigma MapReduce para posteriormente poder ejecutarlos en Hadoop.
- **Spark:** interfaz de programación que permite al usuario programar tanto en Java, Scala, R como en Python y facilita la creación de rutinas que pueden ser lanzadas posteriormente con Hadoop. Incluye librerías específicas para trabajar con facilidad en los campos de: Extract, Transform and Load (ETL), Machine Learning, Graph Computation y Stream Processing. Debido a la importancia de esta herramienta en el desarrollo del presente proyecto, se describirá en la siguiente sección (véase 2.3.2.5).
- **Ambari:** una herramienta web para la gestión y monitorización de clusters que montan Hadoop y que incluye soporte para el sistema de gestión de ficheros HDFS, MapReduce, Hive y Pig entre otros.

Hadoop no es el único framework para Big Data ya que hay otros sistemas distribuidos. Sin embargo, son más específicos y casi todos orientados a aplicaciones en tiempo real. Destacan los siguientes proyectos específicos:

- **Apache Flink:** es un framework especialmente concebido para trabajar sobre flujos de datos. Puede realizar procesamiento por lotes aunque está orientado más a aplicaciones en Real-time. Sus principales características son altas prestaciones y baja latencia, altamente tolerante a fallos a través de la técnica de Lightwighted Distributed Snapshots [34].
- **Apache Storm:** es un framework altamente escalable y adaptado para procesos Real-time siendo sus principales características su rapidez (capaz de trabajar sobre un millón de mensajes de 100 bytes por nodo y por segundo), resistente a fallos y fiable. No tiene capacidad para procesamiento por lotes.
- **Apache Samza:** es otro framework que hace uso de YARN para la gestión de recursos y de Apache Kafka para computación asíncrona cuasi Real-time sobre flujo de datos. Es altamente escalable y tolerante a fallos.

2.3.2.5 Apache Spark

Como hemos visto, Apache Spark [3] es un framework de reciente implementación que pertenece al ecosistema de Hadoop para el análisis de Big Data y que aporta mejoras en cuanto a eficiencia y usabilidad. Es un proyecto de código libre desarrollado inicialmente por la Universidad de Berkeley y donado posteriormente a la fundación Apache. Consiste en una interfaz que permite programar con facilidad aplicaciones que se pueden ejecutar en sistemas distribuidos y tolerantes a fallos, orientado principalmente hacia el campo de Machine Learning.

Nació en los laboratorio AMPLab de la Universidad de Berkeley como un sistema de gestión de computación en clúster que fuera capaz de soportar diferentes métodos de programación distribuida.
Hadoop se había creado para realizar procesamiento en lotes de grandes cantidades de datos, pero no estaba optimizado para procesos que implicasen interactividad e iteración, típicos de los algoritmos que se pueden encontrar en el campo del Aprendizaje Automático.

Una de las principales características de Spark es que emplea un motor de ejecución avanzado de tipo DAG (Directed Acyclic Graph) generando un grafo acíclico dirigido de múltiples etapas de trabajo por cada tarea lanzada en un clúster; mientras que las macros MapReduce de Hadoop generan un DAG de dos

etapas (Map y Reduce). Además, MapReduce necesita guardar en disco los resultados entre las etapas, mientras que Spark permite ejecutar in-memory ahorrando tiempo de escritura y lectura.

Las características que hacen de Spark una herramienta más fácil de usar son su interfaz que permite programar en Java, en R y en Python además de Scala, que es el lenguaje nativo del framework, contando con más de 80 operadores y unas librerías específicas para cada campo en el que se quiera trabajar.

El tipo de dato sobre el que se basa Spark es el RDD (Resilent Distributed Dataset), un tipo de dato de solo lectura, que está distribuido en un conjunto de máquinas del clúster y que, en caso de pérdida de uno de los nodos, puede ser reconstruido. Estos objetos Scala pueden proceder de un sistema de ficheros distribuido, como puede ser Hadoop Distributed File System (HDFS), de la paralelización de un array en Scala, una lista en Python o cualquier otra colección de datos. Pueden generarse también mediante transformaciones de otros RDD o cambiando la persistencia en memoria de otro RDD.

Por defecto los RDD se consideran tipos de datos efímeros y '*perezosos*' (lazy en la literatura), por lo que las particiones de datos se materializan a petición del programa mientras se aplican operaciones sobre ellos y su presencia en memoria es desechada tras el uso. Spark permite al usuario alterar la persistencia de un RDD de dos formas: aplicando un almacenamiento en cache que permite mantener el dato en memoria, para reutilizarlo posteriormente, o guardando el dato en el HDFS, por ejemplo.

En Spark se permiten dos tipos de operaciones en paralelo sobre los RDD: las transformaciones y las acciones. Las primeras implican que a partir de un Dataset Distribuido se genera otro RDD sobre el que se realiza una determinada función. En el caso de las acciones, se aplica una función sobre el conjunto de datos del RDD y se obtiene como resultado un valor nuevo, normalmente único, que se envía al driver. Todas las transformaciones en Spark son '*perezosas*', lo que implica que no se calcula físicamente el valor resultante de una transformación hasta que se requiera. Spark acumula todas las transformaciones que se han solicitado y cuando hay una petición de devolución de un resultado, como puede ser un almacenamiento en disco, se calculan las transformaciones pertinentes a aplicar sobre el RDD.

Para mostrar las mejoras que aporta Spark con respecto a Hadoop para determinados algoritmos interactivos y que producen muchas iteraciones, los

creadores de Spark [35] han llevado a cabo unos experimentos. En concreto, han analizado el comportamiento de ambos frameworks a la hora de realizar una regresión logística, que se usa en estadística para predecir el resultado de una variable categórica. Para este experimento, han empleado un corpus de 29 Gigabytes sobre 20 servidores AWS de Amazon *m1.xlarge* de 4 núcleos y 15 GiB de RAM. En Hadoop cada iteración del algoritmo tarda unos 127 segundos y son todas iguales en tiempo, ya que cada ciclo de repetición del bucle requiere un proceso independiente de MapReduce. Spark tarda 174 segundos para la primera iteración (probablemente debido al uso de Scala en lugar de Java), pero a partir de la segunda, las sucesivas son de 6 segundos; por lo que para 30 iteraciones del algoritmo, Spark tarda 348 segundos en lugar de los 3810 segundos que emplea Hadoop (véase Figura 2.2).

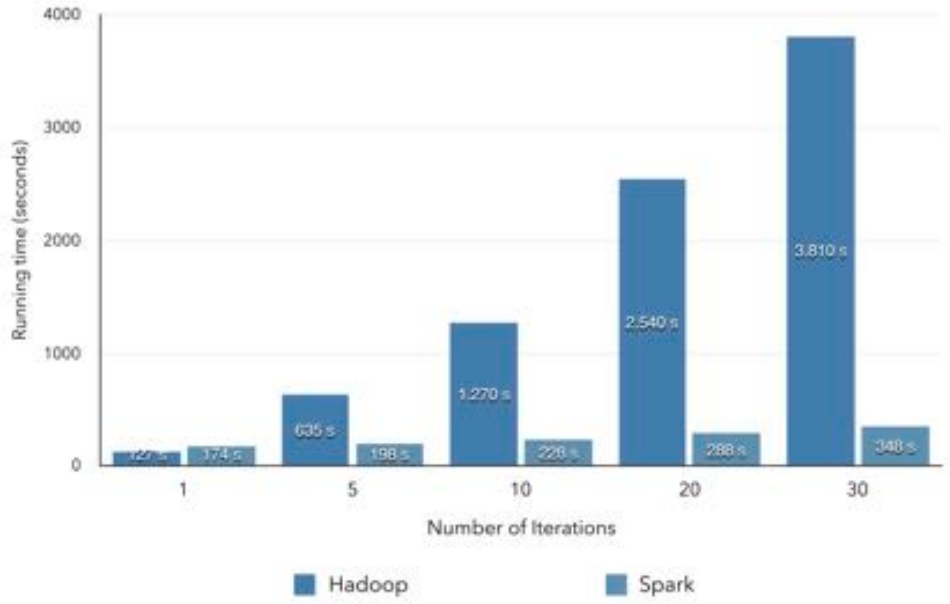


Figura 2.2: Tiempos de ejecución para una función de regresión logística en Hadoop y Spark.

Otro ejemplo de mejora de rendimiento se puede ver cuando se usa la herramienta de forma interactiva y para ello, el mismo equipo de investigación, ha realizado un volcado de 40 Gigabytes de información de Wikipedia. Tras la primera consulta sobre el dataset, la respuesta de Spark tardó unos 35 segundos; mientras que las siguientes búsquedas en los datos se redujeron drásticamente a los 750 ms de media.

Esta herramienta, que ha sido publicada en 2014, está gozando recientemente de buena aceptación en las empresas e instituciones, debido a su sencillez a la hora de integrarla en clústeres preexistentes y que además desempeña eficazmente

tareas de Big Data centradas en Machine Learning, ETL, Real Time Analysis o Graph Computation. De hecho, no sorprende que esté siendo empleada por empresas como Amazon, Autodesk, eBay, Samsung, IBM, Yahoo!, entre otras.

2.3.2.6 Conclusiones

Apache Hadoop es un framework muy empleado en el mundo de Big Data, debido a su gran capacidad de procesamiento por lotes, por lo que es ideal para realizar operaciones simples sobre conjuntos muy grandes de datos o ficheros tan grandes que cualquier otro sistema no podría ni siquiera abrirlos. Además, presenta un ecosistema de aplicaciones que conviven alrededor de esta plataforma que permite gestionar de forma rápida y eficaz el clúster de Hadoop, así como organizar los datos en bases de datos NoSQL o estructurarlos en tablas muy grandes, lo cual sería imposible en otros sistemas.

Por último, se ha visto cómo Apache Spark es un framework que puede servir de apoyo a Hadoop o bien usarse como alternativa. Presenta unas pequeñas variaciones en sus técnicas de análisis que permiten reducir los tiempos de ejecución, especialmente en algoritmos interactivos e iterativos, por lo que resulta un buen compromiso para aplicaciones que hacen uso de grandes cantidades de datos y algoritmos real-time. Su enfoque es más hacia campos concretos de la informática como Machine Learning, ETL o Graph Computation y, en general, se utiliza para desarrollar algoritmos complejos que requieran más pasos que Map y Reduce de Hadoop.

Capítulo 3

Recursos

En esta capítulo se describirán brevemente las colecciones de documentos que componen las bases de datos sobre las que se han generado los modelos LDA (sección 3.1) y los equipos necesarios para la ejecución del código de este proyecto (sección 3.2).

3.1 Bases de datos

Para la realización de las diferentes etapas del proyecto se han empleado distintos corpus de datos. Para una primera fase, se han utilizado colecciones de documentos procedentes de Wikipedia, mientras que para las pruebas sobre Hadoop se ha adoptado el corpus de Associated Press *'20newsgroup'* que incluye 2246 documentos de noticias de dicha organización [36]. Este corpus es el mismo que fue empleado en el proyecto de David Blei, LDA en C de 2003 [30].

La calidad del corpus es crucial a la hora de realizar cualquier tipo de análisis sobre texto, ya que si contiene errores no se podrán generar correctamente los modelos.

Además, como propósito específico de este proyecto se desea poder implementar un análisis empleando un entorno específico de Big Data, por lo que es importante que el corpus sea grande.

Para el desarrollo de Topic Modeling en el entorno Apache Spark, que forma el núcleo central de este trabajo, se ha decidido emplear un corpus de entrenamiento procedente de noticias del periódico español El País del año 2014. Dicho corpus está compuesto por 50.500 ficheros de texto plano con un peso medio de aproximadamente 509 bytes, separado en origen en 7 temas o secciones del diario.

.

Los temas en los que se divide la base de datos de entrenamiento son:

- **Cultura**
- **Economía**
- **Internacional**
- **Nacional**
- **Sociedad**
- **Deportes**
- **Tecnología**

En primera instancia, se ha decidido borrar todos los ficheros que por peso no fueran significativos. En concreto se han eliminado del corpus todos aquellos ficheros cuyo peso fuera menor de 200 bytes y no fueran representativos para el análisis. Tras esta primera limpieza se ha obtenido un corpus de 45.940 ficheros de entrenamiento, distribuidos en los 7 temas del periódico de la siguiente manera:

	Cultura	Economía	Deportes	Internacional	Nacional	Sociedad	Tecnología	Total
Número ficheros	1.796	8.472	5.960	13.200	7.800	5.200	3.512	45.940
Tamaño	921 KB	4,4 Mb	3,9 Mb	7,4 Mb	4,3 Mb	3 Mb	1,5 Mb	25,3 Mb

Tabla 3.1: Números de ficheros y tamaño de las secciones del corpus.

Para las pruebas del Capítulo 5 se ha empleado un pequeño corpus de test, procedente de las secciones de Cultura, Economía, Internacional, Nacional, Sociedad, Deportes y Tecnología, también del periódico El País, obtenidas en Marzo de 2016.

3.2 Servidor

Para la realización de este proyecto se ha empleado un servidor HP ProLiant DL380 G5 de 4 núcleos Intel Xeon 64 bits de 3.1GHz con 10 Gigabytes de RAM a 1333 MHz ECC registrada y 8 discos duros en raid 1 de 500 Gigabytes [37]. Sobre dicho servidor se ha instalado el sistema operativo LINUX Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-35-generic x86_64). En el servidor se ha instalado el entorno Apache Spark en su última versión estable, 1.6.1 con pre-build para Apache Hadoop 2.6. El servidor ha sido amablemente cedido por la empresa Sigma Technologies para la realización de este Proyecto de Fin de Carrera.

Capítulo 4

Generación de modelos

En este capítulo se detallan las pruebas que se han realizado sobre distintas herramientas y frameworks, empezando por entornos no distribuidos como Mallet (sección 4.2) y terminando con sistemas distribuidos como Apache Hadoop (sección 4.3) y Apache Spark (sección 4.4).

4.1 Introducción

Aunque el objetivo final era, desde un principio, desarrollar un generador de modelos LDA sobre un sistema distribuido, se han realizado unas primeras pruebas en Mallet, para comprobar el funcionamiento del algoritmo y acercarse a los conceptos subyacentes.

Las pruebas sobre Apache Hadoop han sido algo más complejas ya que la instalación del framework ha sido laboriosa. El lenguaje Java y los paradigmas MapReduce han resultado poco amigables al comienzo, pero han permitido la familiarización con la programación distribuida.

En la última sección se expondrán los pasos seguidos para la realización de un sistema de detección de estructuras latentes de temas, empleando el algoritmo Latent Dirichlet Allocation LDA, en Apache Spark. Se detallará la metodología y la implementación necesaria para desarrollar el módulo generador LDA en Scala y se mostrarán los resultados de una forma gráfica.

4.2 Primeras pruebas: Mallet

En una primera fase del proyecto se ha decidido comprobar la factibilidad de implementar un sistema básico de inferencia de temas como Mallet, que hace uso del algoritmo de Latent Dirichlet Allocation sobre un sistema no distribuido

basado en Java, y que además utiliza un algoritmo de inferencia basado en muestreo, como es el algoritmo de Gibbs Sampling. Esta herramienta ha sido diseñada por la Universidad de Massachutes y es bastante ilustrativa en cuanto al funcionamiento del algoritmo. Además permite generar estructuras temáticas latentes sobre una colección de datos por consola de comandos y de forma bastante sencilla. El primer problema que nos hemos encontrado al implementar esta herramienta sobre un sistema no distribuido, ha sido la dificultad de trabajar con una gran cantidad de ficheros ya que, incluso para pequeñas colecciones de documentos, la herramienta tarda bastante en ejecutarse.

En estudios académicos se ha empleado Mallet con corpus de hasta 200.000 ficheros, pero el problema principal que surge es la cantidad de memoria que esta aplicación consume, por lo que conforme aumentamos el número de datos, necesitaremos aumentar linealmente la memoria requerida por el sistema para procesar la información.

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
```

Figura 4.1: Mensaje de error de espacio en memoria.

Para poder resolver este problema habría que clusterizar el programa para ejecutarlo en un sistema distribuido, pero antes de tener que recurrir a esta tarea, se han intentado otras alternativas que nos permitan diseñar una implementación del algoritmo LDA sobre un sistema distribuido.

4.3 Implementaciones de LDA en Apache Hadoop

Una de las herramientas distribuidas por antonomasia para el análisis de gran cantidad de datos es, como se ha visto en la sección 2.3.2, Apache Hadoop. Esta última es de gran utilidad para poder aplicar algoritmos costosos computacionalmente, tanto por su complejidad como por trabajar con muchos ficheros. Se ha instalado esta herramienta en el servidor, descrito en la sección 3.2, que se empleará para estas pruebas. Para ello se necesita un sistema operativo GNU/LINUX, con máquina virtual Java SE 7 o superior y se empleará la versión 2.7.1 [38]. He de precisar en este punto, que la instalación de este framework no ha sido nada fácil, ya que implica muchos conocimientos de informática, de redes y del sistema GNU/LINUX. En concreto, implica la instalación de dependencias y la inicialización de variables de entorno y ficheros de configuración de Hadoop.

Una vez instalado el framework, se han analizado las herramientas disponibles para Hadoop que simplifiquen el desarrollo de un sistema de análisis de temas utilizando el algoritmo LDA. Una de las primeras pruebas, se hizo con la herramienta Hadoop-Lda [39] para poder obtener cierto grado de familiarización con la programación distribuida. Esta aplicación escrita en lenguaje Java es una implementación de LDA a través del algoritmo inferencial de Gibbs Sampling, haciendo uso de las macros MapReduce.

Esta implementación básica de LDA permite generar modelos con bastante sencillez y además, en caso de fallo, hecho bastante común en los sistemas en clúster, permite retomar la simulación desde la última iteración correcta. La salida que genera, es un fichero de texto que contiene los principales parámetros de la simulación, como los hiperparámetros α y η, el número de temas k y el tamaño del diccionario $VocabSize$, así como todas las palabras que lo forman, junto con la frecuencia de ocurrencia de cada una de éstas dentro de cada tema.

```
$HADOOP jar hadooplda-hadoop.jar train \
--input=/users/datainput --output=/users/datainput.lda.model \
--working_dir=/users/datainput.lda.training \
--num_topics=128 --num_iterations=100 \
--iterations_to_keep=2 --max_num_words=50000 \
--input_format=text --min_df=5 \
--alpha=0.45 --beta=0.01
```

Figura 4.2: Ejemplo de manejo por consola de HadoopLda.

Esta herramienta permite poner en funcionamiento el sistema en poco tiempo, y podría considerarse como una implementación parecida a Mallet pero, en este caso, para un sistema distribuido.

Sin embargo, el inconveniente principal de Hadoop-Lda es que no es una aplicación desarrollada para ofrecer una solución integral para la generación de modelos, implicando que todos los procesos previos habría que desarrollarlos por separado.

En una segunda prueba, se decidió instalar la herramienta Mr.LDA, desarrollada por Ke Zhai y basada en el artículo homónimo [40]. Este proyecto se centra en la creación de un sistema de detección de temas a gran escala, flexible, distribuido y que hace uso, a diferencia de los citados en anteriores secciones, de un algoritmo inferencial variacional. Como se puede ver en la tabla 4.1, Mr.LDA hace uso del paradigma MapReduce para Hadoop y está diseñada en Java.

Application	Framework	Inference	Likelihood Computation	Asymmetric α Prior	Hyperparameter Optimization	Informed β Prior	Multilingual
Mallet [1]	Multi-thread	Gibbs	✓	✓	✓		✓
GPU-LDA [43]	GPU	Gibbs & V.B.	✓				
Async-LDA [44]	Multi-thread	Gibbs	✓		✓		
pLDA [45]	MPI & MapReduce	Gibbs					
YiLDA [46]	Hadoop	Gibbs	✓	✓	✓		
Mahout [47]	MapReduce	V.B.	✓				
Mr.LDA [40]	MapReduce	V.B.	✓	✓	✓	✓	✓

Tabla 4.1: Comparación entre distintas aplicaciones, framework sobre el que se ejecuta y el algoritmo inferencial que emplea.

Aunque esta herramienta es más completa que las anteriores, sigue siendo un sistema no adaptado para poder programar módulos propios y modificar tanto el corpus como los procesos de simulación, para obtener sistemas más complejos. Para poder hacer muchas de las operaciones previas al procesado con el algoritmo de LDA, se tiene que recurrir a comandos de Shell de LINUX que hacen el desarrollo y el procesado algo más lento.

He de añadir que para estas pruebas sobre Apache Hadoop, el principal escoyo ha sido la falta de documentación en la literatura y en la red. Los errores que generan las funciones MapReduce que se aplican para el LDA no están suficientemente documentados, por lo que resulta imposible averiguar el origen de los fallos. Además, cuando se produce una excepción en el thread de ejecución de Java, Hadoop siendo un sistema altamente tolerante a fallos, notifica al usuario de ello, pero sigue ejecutando el código hasta terminar con las tareas que tiene pendientes. Esto dificulta muchísimo las simulaciones, sobre todo si no se está acostumbrado a trabajar en este tipo de sistemas tolerantes a fallos y distribuidos o en entornos Java que son muy verbosos.

Por último, para poder programar con algo más de agilidad con respecto al framework anterior, se planteó desarrollar el sistema sobre Apache Spark. Como se ha visto con anterioridad (sección 2.3.2.5), éste es un framework independiente que también funciona como interfaz de programación para Hadoop y que además incluye librerías específicas para Aprendizaje Automático. Para más detalles sobre la implementación sobre este framework véase la siguiente sección 4.4.

4.4 Implementación de LDA en Apache Spark

Tras unas primeras pruebas realizadas con diferentes herramientas, descritas en las secciones anteriores y en diferentes entornos, se ha decidido escoger un sistema distribuido como Hadoop Spark ya que ofrece numerosas ventajas. Dentro de ellas destacan la fácil programación, debido a su interfaz programable tanto en R, Python, Java como en Scala, y la disponibilidad de librerías que facilitan aún más el desarrollo de código. En concreto, la librería **MLlib** específicamente diseñada para Machine Learning es una herramienta muy valiosa a la hora de realizar cualquier análisis en el campo del Aprendizaje Automático.

4.4.1 Instalación del framework

Como primer paso se ha procedido a instalar Apache Spark 1.6.0 en el servidor con sistema operativo GNU/Linux y con máquina virtual de Java SE 7. Como puntualización, Spark necesita Hadoop para leer datos distribuidos en sistemas tipo HDFS, pero actúa como herramienta independiente a la hora de ejecutar código. Por ello la instalación se efectúa a través de un pre-build específico para una determinada versión de Hadoop. Sorprendentemente la instalación de Apache Spark ha sido sencilla, sobre todo si se compara con el anterior framework, ya que ha sido suficiente descargar el código y empezar a trabajar con ella.

Así mismo, esta herramienta se puede integrar con la plataforma Anaconda [41] que confiere una interfaz web amigable para poder programar en Python y prescindir de la consola. Es útil precisar que a diferencia de Java, Pyhton es un lenguaje interpretado y de código bastante más legible, por lo que es ideal para poder empezar a trabajar con los paradigmas de programación de los sistemas distribuidos, que resultan algo más complejos con respecto a los convencionales.

La documentación del framework es bastante completa y permite con facilidad entender los conceptos básicos y necesarios para utilizar los tipos de datos, las transformaciones y las acciones que se pueden aplicar sobre éstos. Sin embargo, la documentación relativa a las funciones implementadas en las librerías como MLlib no es tan completa y se requiere de más información, que por otro lado no es fácil conseguir.

Como se verá en el desarrollo posterior, tras unas primeras pruebas en Python, se ha decidido continuar el proyecto programando en lenguaje Scala, que es el lenguaje nativo de la plataforma. Esta decisión se ha tomado debido a que la

documentación de la librería MLlib para Python no es exhaustiva y que para algunas funciones específicas, como **CountVectorizer** (véase sección 4.4.2.3), no hay una implementación en la API de Python de Spark.

Hay que tener en cuenta que la implementación de LDA en la librería MLlib ha sido incluida recientemente en Spark (a partir de la versión 1.3.0 en marzo de 2015) y a día de hoy (versión 1.6.1 de marzo de 2016), sigue siendo una función experimental en desarrollo activo, por lo que no está completa y tiene amplios márgenes de mejora. A pesar de estos pequeños inconvenientes, las ventajas de emplear una plataforma como Apache Spark siguen siendo muchas, por lo que se ha decidido proseguir con la implementación del sistema en dicho framework.

4.4.2 Proceso de generación de un modelo LDA

En esta sección se detallan todos los pasos que se han empleado para generar un modelo de estructuras temáticas latentes empleando el algoritmo LDA en Apache Spark. En la figura 4.3 se muestra un diagrama de bloques del proceso de generación de modelos.

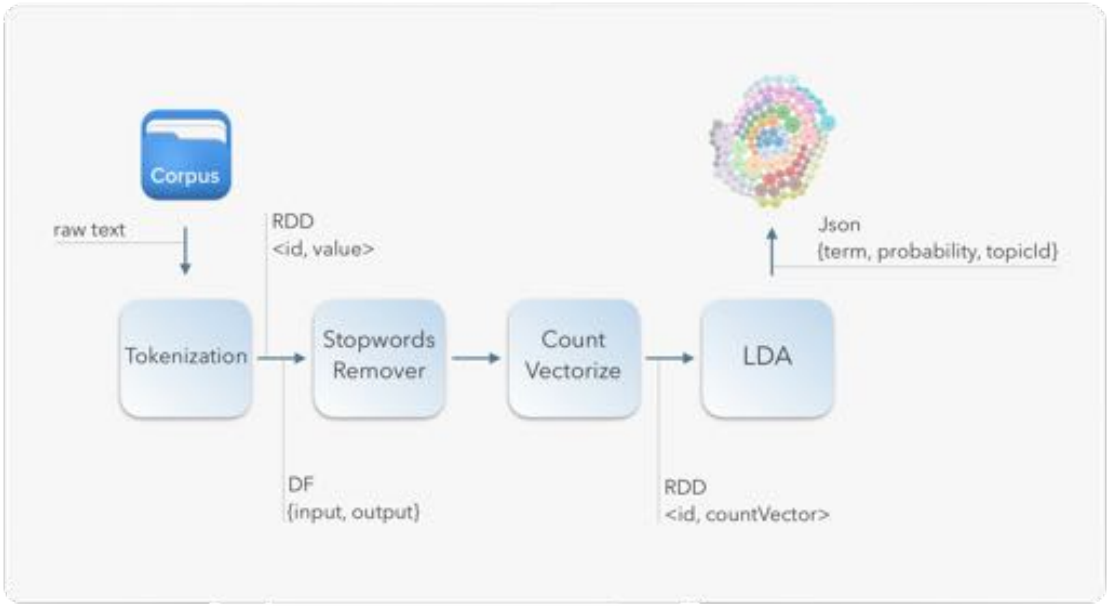


Figura 4.3: Diagrama de Proceso de generación de un modelo LDA.

En primer lugar hay que aplicar un proceso de Tokenización para separar las distintas palabras que constituyen el documento (4.4.2.1); en un segundo momento hay que eliminar las palabras Stopwords (4.4.2.2); a continuación hay que crear el diccionario y generar la matriz de ocurrencia de palabras del diccionario dentro del corpus (4.4.2.3). Por último, se aplica sobre dicha matriz el

algoritmo LDA y se procesa la salida para poderla representar en forma de gráfica clusterizada (4.4.2.4).

4.4.2.1 Tokenización

El primer paso para la creación del diccionario consiste en el proceso de tokenización del corpus que se basa en una segmentación del texto, para extraer las distintas palabras y poder operar con ellas.

Esta operación aparentemente sencilla, conlleva una serie de decisiones que afectan a los resultados finales de la tarea. En primer lugar, en la literatura hay muchos ejemplos de tokenización de textos en inglés, pero no hay muchos en otros idiomas. El español incluye muchos símbolos específicos: las tildes, las diéresis, la virgulilla de la ñ entre otros, que no se contemplan en la mayoría de las funciones usadas, por lo que no se adaptan al abecedario característico del castellano.

En una primera implementación en lenguaje Python se ha creado una función específica aplicada a una macro Map para realizar un filtrado manual y admitir las palabras que incluyen dichos símbolos.

En una segunda y definitiva implementación en lenguaje Scala se ha decidido hacer uso de la librería MLlib para Machine Learning de Hadoop Spark. A continuación se detallarán los pasos seguidos para la implementación, tal como se recoge en el diagrama de bloques de la figura 4.4.

A través de la función **wholeTextFiles** se lee desde el directorio en el que se encuentra el corpus y se obtiene un RDD consistente en una tupla $< key, value >$, donde el valor representa el contenido de cada fichero. A continuación, se trasforma el RDD en un DataFrame (DF) y, sobre la columna que contiene el contenido de los documentos, se aplica la función **RegexTokenizer**.

Esta función nos permite seleccionar la entrada, la salida, la longitud mínima de palabra que queremos analizar y el patrón regexp que queremos usar para trocear el documento e individuar las palabras. Este enfoque permite centrarnos en el delimitador que separa las distintas palabras y admitir cualquier símbolo que haya entre tokens, incluyendo los símbolos específicos del español.

En cuanto a la longitud mínima de las palabras, tras una serie de experimentos, se ha decidido inicializarlo con valor 4, permitiendo descartar la mayoría de siglas,

palabras pequeñas y números, sin descartar fechas ya que podrían ser interesantes para el análisis posterior.

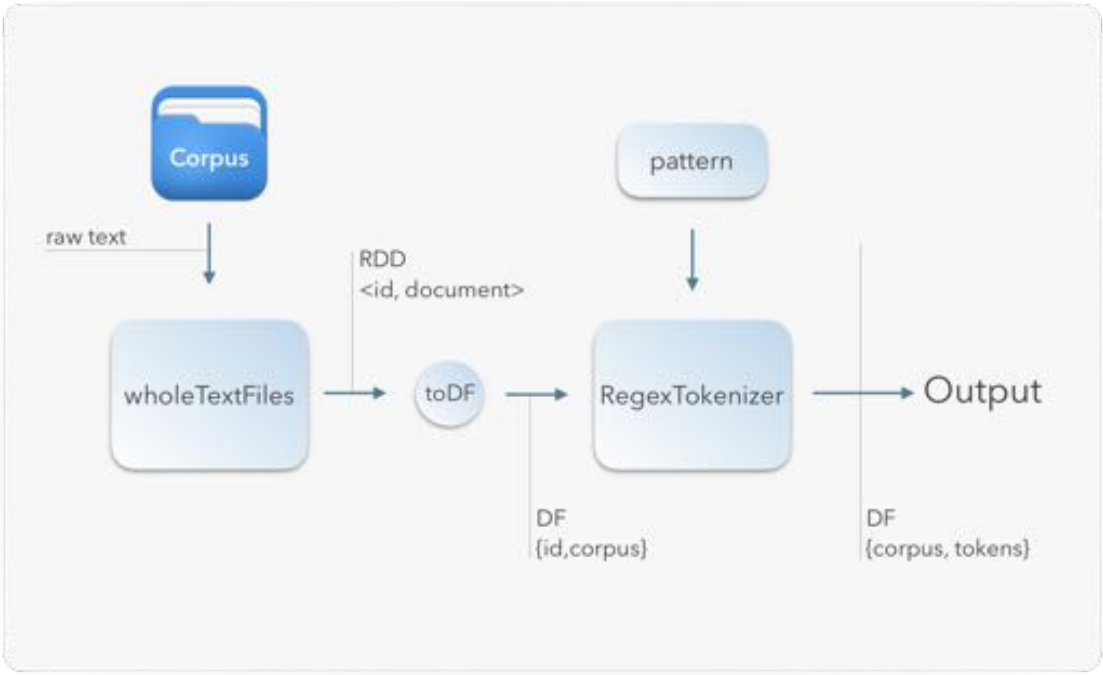


Figura 4.4: Diagrama de entrada/salida del módulo de tokenización.

En cuanto al patrón de búsqueda, se ha empleado la siguiente expresión regular en Scala para separar los distintos tokens:

("(\\s+)|(\\. +)|(\\, +)|(\\? +)|(\\!+)|(\\\"+)|(\\: +)|(\\; +)|(—)|(¿)|(¡)|(”)|(\\()|(\\))")

Los tokens empleados para separar las palabras del corpus, se han obtenido tras repetidas pruebas, ya que a cada simulación iban apareciendo nuevos símbolos que quedaban sin detectar.

Se ha decidido también transformar todo el corpus en minúscula para simplificar el análisis y los diccionarios.

4.4.2.2 Stopwords

Como se ha visto en la sección 2.2.2, uno de los principales problemas a la hora de contar el número de palabras y la frecuencia de aparición de las mismas en cada documento, es el fenómeno de la aparición de stopwords. Éstas son palabras que se emplean mucho en cada lengua y que no son de particular interés para el análisis de temas, ya que carecen de significado léxico. Suelen ser palabras de

significado gramatical o abreviaturas que hay que eliminar del texto antes de crear el diccionario.

En el primer diseño del sistema, que se realizó en Python, se planteó un filtrado a través de una función Map que comprobaba que cada token del corpus no perteneciera a la lista de stopwords. Esta implementación es poco eficaz ya que no aprovecha del todo la programación distribuida, por lo que en la siguiente implementación se ha decidido hacer uso de las librerías MLlib e implementar la función **StopWordRemover()**. Dicha función recibe como argumento, además de los valores de las columnas de entrada y de la salida, el dataset distribuido RDD donde se han leído todas las palabras que se deberán excluir. Para un diagrama del funcionamiento de este módulo véase figura 4.5.

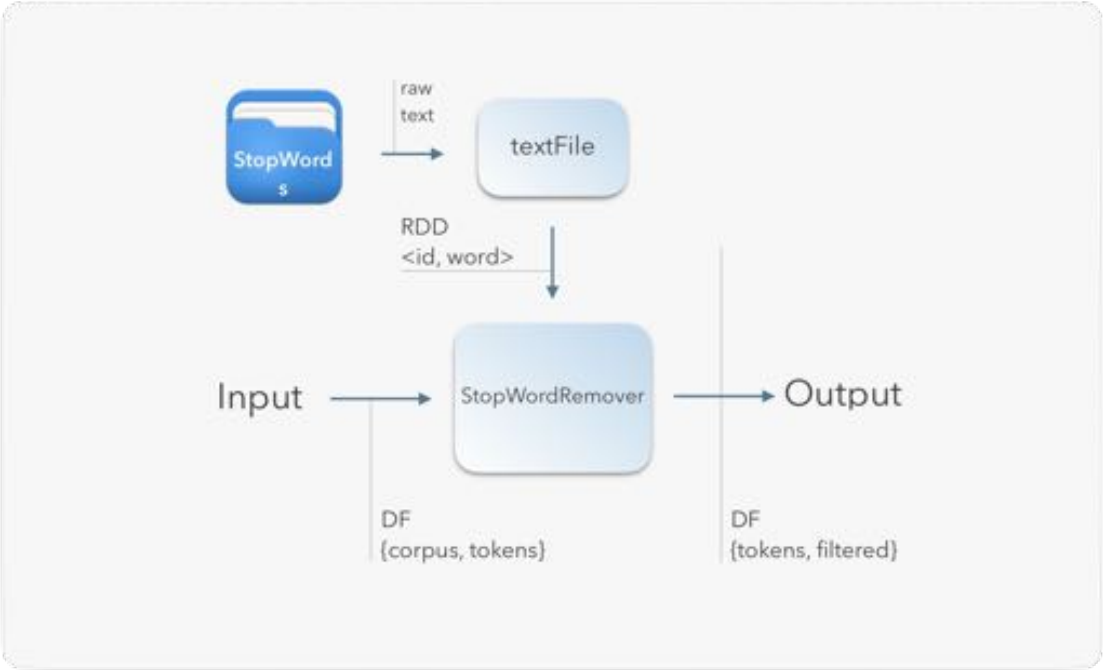


Figura 4.5: Diagrama de entrada/salida del módulo de filtrado de stopwords.

La creación de la colección de stopwords se ha tenido que hacer desde cero y con cada simulación en la que aparecían palabras sin mucho significado léxico, la lista de dichas palabras se ha ido ampliando. La lista completa de palabras empleada para las simulaciones se incluye en el *Apéndice A*.

4.4.2.3 Creación del diccionario

En la siguiente etapa de la realización de este proyecto, se ha procedido a crear la matriz de diccionario de palabras a partir del corpus, que en la literatura recibe el

nombre de bag of words. Dicha matriz está constituida por la frecuencia de aparición de cada token del diccionario en cada documento del corpus.

Emplearemos la función **CountVectorize()** para crear un diccionario, ya que no disponemos de uno a priori. Esta función recibe como argumentos, además del nombre de la columna de entrada y de salida del DataFrame donde se aplicará la trasformación, el tamaño del diccionario, el número mínimo de apariciones de un token para incluirlo en el diccionario y el número mínimo de documentos en los que tiene que aparecer el token, véase figura 4.6.

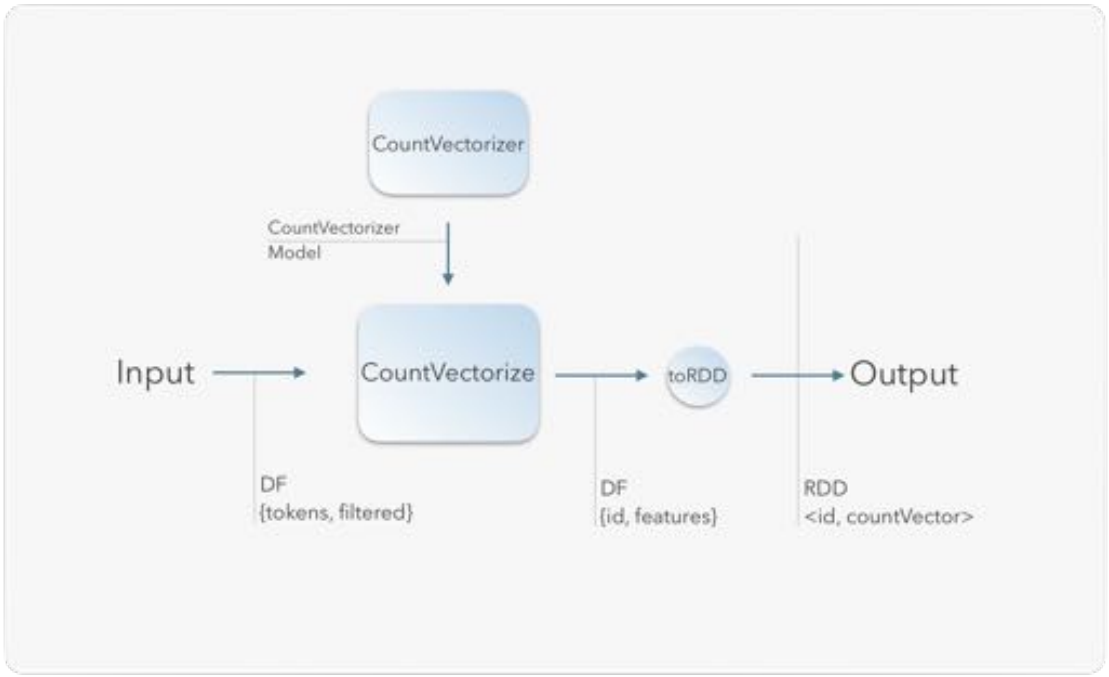


Figura 4.6: Diagrama de entrada/salida del módulo de creación de Diccionario.

El diccionario se creará a partir de las primeras $VocabSize$ palabras que aparecen con más frecuencia en el corpus, donde $VocabSize$ es el tamaño que se elige para el diccionario.

Se ha decidido elegir $VocabSize = 2000$ para el diccionario y 3 para el mínimo de documentos en los que tiene que aparecer la palabra para ser incluida, mientras que se ha decidido dejar en 0 el número mínimo de aparición de palabras por documento.

Si se realiza la tokenización y el filtrado de stopwords para un corpus de dos documentos como el que se muestra en la figura 4.7, el resultado se puede ver en la figura 4.8.

```
El gato o gato doméstico, Felis silvestris catus.
El gato doméstico es una de las mascotas más populares.
```

Figura 4.7: Ejemplo de corpus de dos ficheros.

id	texts
0	Array("gato", "gato", "doméstico", "felis", "silvestris, "catus")
1	Array("gato", "doméstico", "mascotas", "populares")

Figura 4.8: Tokenización del corpus y filtrado de stopwords.

Un ejemplo del funcionamiento de **CountVectorize()**, para VocabSize igual a 4, podría ser el que se muestra en la figura 4.9.

id	texts	vector
0	Array("catus","doméstico","felis","gato")	(4,[0,1,2,3],[1,1,1,2])
1	Array("gato", "doméstico")	(4,[0,1,2,3],[0,1,0,1])

Figura 4.9: Resultado tras la aplicación de la función CountVectorize.

Una vez obtenido el vector de apariciones de las palabras del diccionario en cada documento, se procede a transformar el DataFrame en un dataset distribuido de tipo RDD.

4.4.2.4 Generación de modelos genéricos con Latent Dirichlet Allocation

Como se ha comentado en la sección 2.2.3.4, el algoritmo Latent Dirichlet Allocation hace uso de algoritmos de métodos variacionales para calcular la probabilidad marginal o evidence. En la librería MLlib de Spark Apache se permite seleccionar el algoritmo variacional con el que se realiza dicho cálculo. Los posibles algoritmos que se pueden emplear son: **OnlineLDAOptimizer** [49] y **EMLDAOptimizer** [50].

El primero es un algoritmo de inferencia variacional interactivo y bayesiano (Online Variational Bayes), mientras que el segundo genera los clusters de temas empleando el método de Expectation Maximization sobre la función de semejanza. El primero es normalmente de menor impacto para la memoria del

sistema, mientras que el segundo consume más recursos ya que necesita más iteraciones para converger y generar resultados correctos.

El algoritmo de inferencia que se quiere emplear se define a través del constructor **setOptimizer** y genera dos modelos distintos en función de dicho parámetro, para OnlineLDAOptimizer se creará un LocalLDAModel, mientras que EMLDAOptimizer generará una estructura del tipo DistribuitedLDAModel.

Los demás parámetros que admite LDA son el número de temas k, $docConcentration$ y $topicConcentration$ que son los parámetros α y η de concentración de la distribución Dirichlet a priori, $Dir(\alpha)$ y $Dir(\eta)$ que se han analizado en la sección 2.2.3.3, $maxIterations$ que indica el número máximo de pasadas que se permite aplicar a los algoritmos y $checkpointInterval$ que define la frecuencia con la que se crean copias del proceso de iteración y permite reanudar la ejecución justo antes de haberse producido un fallo.

Para cada uno de los dos algoritmos variacionales arriba mencionados, se permiten determinados valores de los parámetros anteriormente descritos y ulteriores parámetros específicos de cada modelo.

En el caso del algoritmo variacional bayesiano interactivo, los parámetros $docConcentration$ y $topicConcentetration$ quedan definidos como:

- $docConcentration$ es un vector asimétrico de valores del hiperparémetro Dirichlet α y por defecto cada elemento es igual a $\alpha = 1/k$;
- $topicConcentration$ es un vector simétrico cuyo valor por defecto también es $\eta = 1/k$;

Además, en este caso específico, $maxIterations$ indica el número de minibatches que se emplearán para la inferencia.

A su vez, para este modelo en concreto, se permite modificar los siguientes parámetros:

- $miniBatchFraction$ que indica la porción de corpus usada en cada iteración.
- $optimizeDocConcentration$ que permite optimizar el hiperparámetro α de la distribución Dirichlet, ya que implementa el algoritmo de Maximum Likelihood en cada pasada para estimar dicho parámetro y emplearlo en la siguiente iteración.
- τ_0 y κ que son los parámetros de la curva de aprendizaje $(\tau_0 + iter)^{-\kappa}$.

En el caso del algoritmo de Expectation Maximization:

- $docConcentration$ es un vector simétrico por lo que todos los valores tendrán que ser iguales, por defecto $\alpha = 1 + 50/k$;
- $topicConcentration$ es también un vector simétrico cuyo valor por defecto es $\eta = 1 + 0.1$.

Ambos modelos producen una matriz $topicsMatrix$ de $VocabSize$ filas por k columnas en la que cada palabra se relaciona con un tema. Sin embargo, a efectos de este proyecto nos interesará más la matriz de temas $describeTopics$ en la que cada tema queda definido como vector de palabras del diccionario ordenadas por peso dentro del mismo tema. Un ejemplo de ello se muestra en la figura 4.10.

Figura 4.10: Ejemplo de temas inferidos a partir de un texto empleando el algoritmo LDA.

Además de estas dos matrices, cada modelo proporciona también otros datos, que veremos a continuación. Para el algoritmo OnlineLDAOptimizer, el LocalLDAModel permite calcular la función de verosimilitud $logLikelihood$ sobre una colección de documentos, partiendo de los temas encontrados en el corpus. También permite calcular la perplejidad del modelo de lenguaje $logPerplexity$ de una colección de documentos a partir de los temas inferidos en el corpus.

Para el algoritmo EMLDAOptimizer se guarda también una matriz $topTopicsPerDocument$ en la que, por cada documento, se detallan los temas encontrados con su peso relativo dentro del texto en concreto. En la matriz $topDocumentPerTopic$ se detallan ordenados por peso los documentos en los que cada tema aparece con más frecuencia.

Este modelo permite calcular también el $logPrior$ que es la probabilidad logarítmica a priori de las distribuciones de los temas inferidos dados los hiperparámetros y, por último, el $logLikelihood$ que es la probabilidad logarítmica de semejanza del corpus.

Uno de los problemas principales a la hora de decidir cuál de los dos algoritmos inferenciales aplicar para este proyecto, ha sido que estas funciones siguen siendo experimentales y en constante desarrollo, por lo que no todas las funcionalidades están disponibles en ambos modelos. Por ejemplo, se permite la conversión de DistribuitedLDAModel a LocalLDAModel, pero no viceversa.

Para este proyecto se han hecho pruebas con ambos modelos y se ha podido constatar que para un mismo corpus de datos, el algoritmo OnlineLDAOptimizer es algo más ligero en memoria y además necesita entre 20 y 40 iteraciones para converger. Para el mismo número de iteraciones el algoritmo EMLDAOptimizer no es capaz de crear un modelo igual de consistente ni de inferir todos los k temas, por lo que muchos resultan ser repeticiones. Para obtener resultados aceptables con este último algoritmo hay que aumentar el número de pasadas a 80, por lo que para la realización de este proyecto se ha decidido utilizar $maxIteration = 100$ para evitar repetir la generación de modelos por falta de iteraciones.

Para una comparativa de tiempos de ejecución entre los dos algoritmos, realizada sobre un corpus de 4,5 millones de artículos de Wikipedia, realizada en un AWS EC2 de 16 nodos 16 r3.2xlarge de 61GiB, véase la figura 4.11 [48].

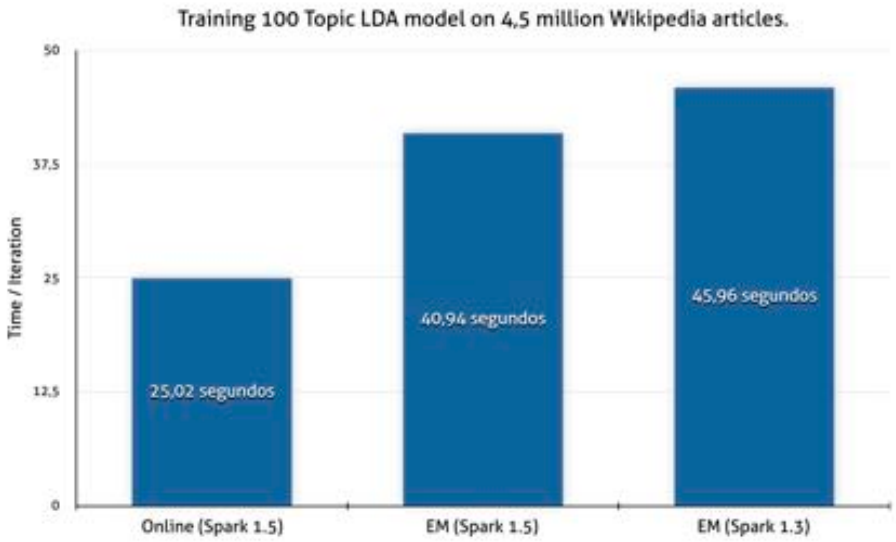


Figura 4.11: Comparativa entre los distintos algoritmos de inferencia para Apache Spark.

Llegados a este punto, hemos procedido a aplicar el algoritmo inferencial OnlineLDAOptimizer sobre todo el corpus, transformando los resultados a Json.

En la figura 4.12 se muestra un esquema funcional del sistema de generación de modelo y de las entradas y salidas del mismo.

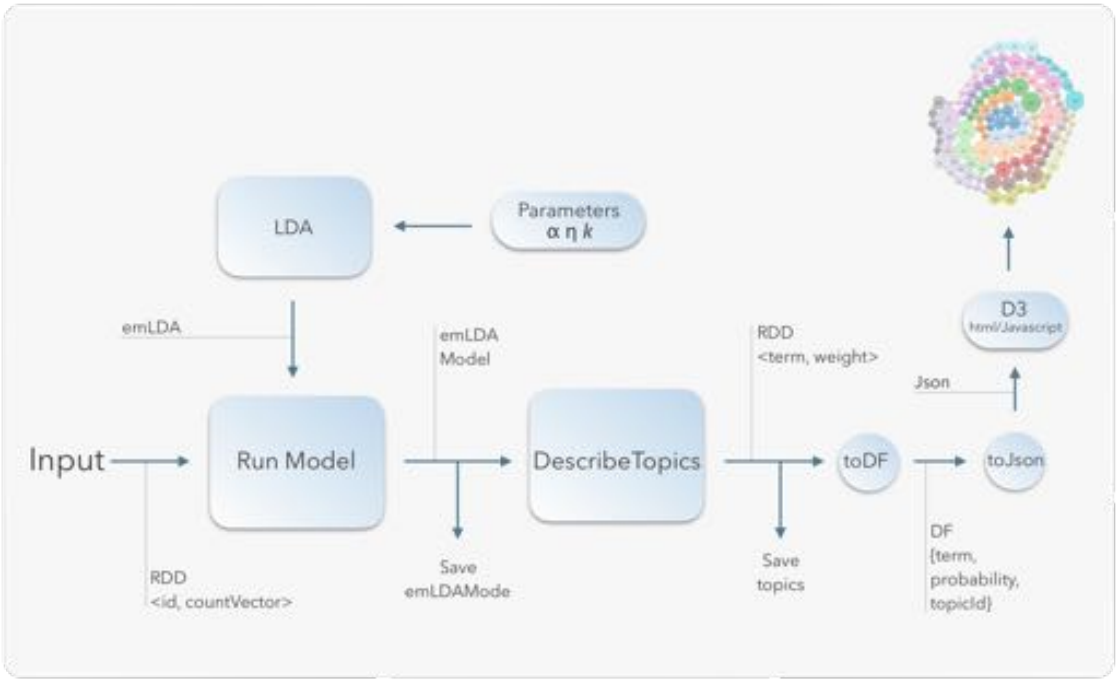


Figura 4.12: Diagrama de entrada/salida de la función de generación de modelos LDA.

A partir de la salida del sistema, se ha diseñado un pequeño script a través de la librería javascript D3 [42] para crear un gráfica con los datos obtenidos. El resultado, como se puede apreciar en la figura 4.13, es una gráfica clusterizada donde cada círculo de un mismo color representa palabras de un mismo tema y el diámetro de cada círculo representa la importancia de la palabra dentro de ese tema.

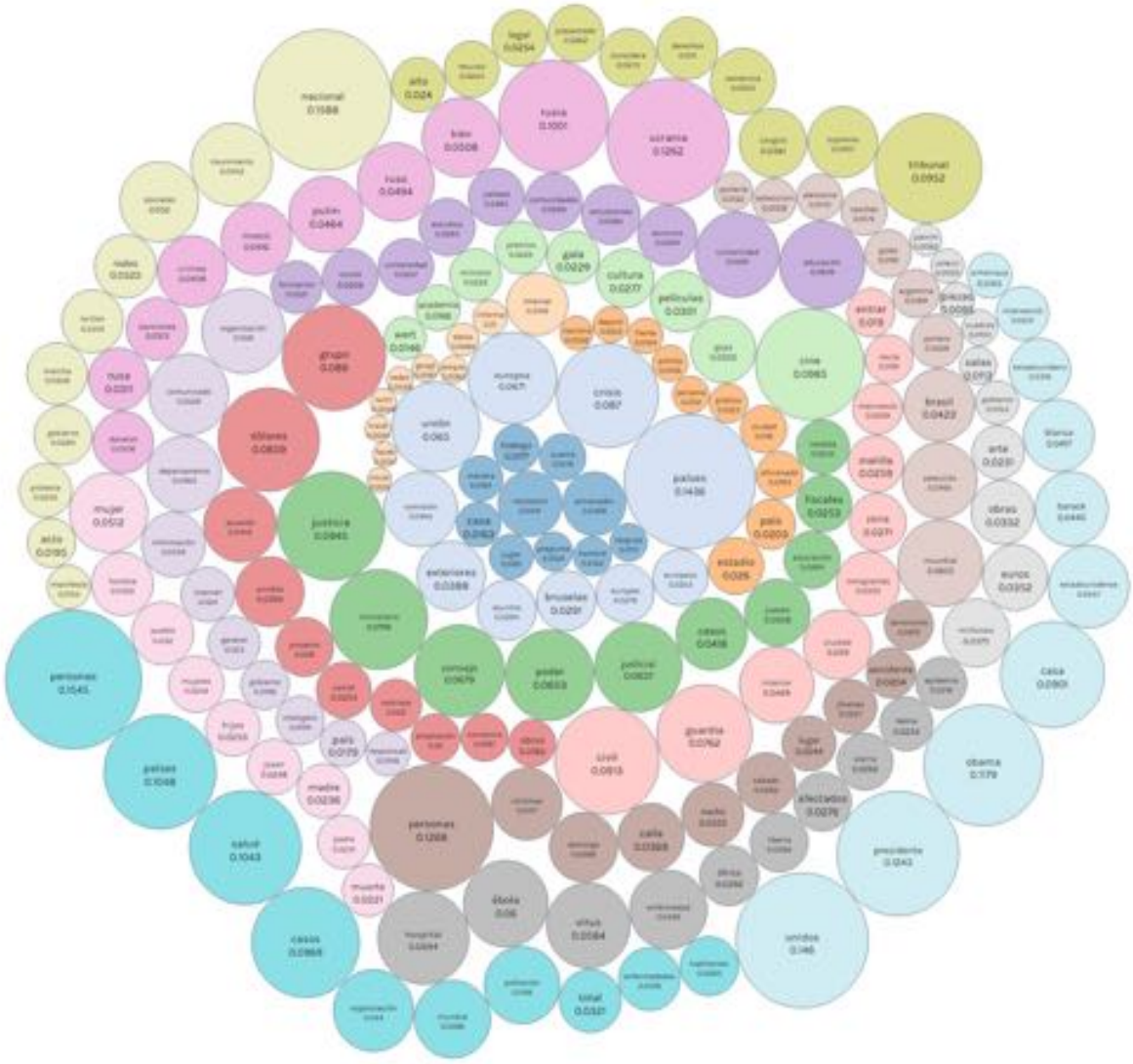


Figura 4.13: Gráfica clusterizada de un modelo LDA para un diccionario de 2000 palabras y una estructura latente de 20 temas.

Capítulo 5

Generación de modelos temáticos y optimización

En este capítulo se hará una breve introducción al sistema que se quiere desarrollar (sección 5.1), se generarán distintos modelos por cada tema (sección 5.2), se compararán con documentos externos al corpus de entrenamiento (sección 5.3), se creará un sistema de optimización de variables parametrizadas (sección 5.4) y por último se analizarán los resultados obtenidos (sección 5.5).

5.1 Introducción

En este capítulo se ha decidido aplicar el algoritmo LDA sobre fracciones de texto previamente clasificadas manualmente, para permitir desarrollar una aplicación capaz de comparar ficheros externos al corpus de entrenamiento, con los modelos de distinta temática. Esto nos permitirá ver cuál es el tema de cada estructura temática en el que el documento encaja mejor. Con este enfoque diferente, se pretende ver si se pueden utilizar diferentes modelos para clasificar noticias de prensa.

Como se ha explicado en la sección 3.1 de este proyecto, los datos pertenecientes al corpus del periódico El País, están clasificados en 7 temas que son: Cultura, Deporte, Economía, Internacional, Nacional, Sociedad y Tecnología. Por lo que, a partir de ahora, en el desarrollo de este sistema se plantea aplicar el algoritmo LDA sobre cada subconjunto temático, de forma que se podrá analizar con más profundidad la distribución de sub-temas dentro de cada tema.

Para evitar confusiones con la terminología, a partir de ahora, se decide emplear el término tema sólo para identificar el tema al que pertenece el documento en objeto, mientras que se empleará el término $subtopic$ para referirse a los temas de la estructura latente inferida a través de LDA sobre la fracción del corpus perteneciente a un determinado tema.

5.2 Proceso de generación de modelos para distintos temas

Para poder realizar este proceso es necesario reorganizar el código, de tal forma que se pueda ejecutar en un bucle en el que se variarán una serie de parámetros. Para cada uno de los 7 temas habrá que generar un modelo a partir de los documentos que componen dicha sección y además habrá que variar dos parámetros esenciales para la generación de modelos estables: $VocabSize$ y $numTopics$ (hasta ahora denotado como k).

El idioma castellano, según la RAE, consta de unos 93.000 vocablos a los cuales hay que añadir un 20% de palabras que normalmente no están recogidas en los diccionarios, por lo que podríamos hablar de un total 110.000 lemas. También hay que considerar que, de media, cada español hace uso de un diccionario muy reducido, que consta de apenas unas 500 palabras, mientras que se estima que un escritor o periodista podría hacer uso de un diccionario aproximado de entre 3.000 y 5.000 palabras. Esto último, nos llevaría a pensar que el vocabulario de cada tema podría variar entre 2.000 y 5.000 palabras, ya que hay que tener en cuenta que las palabras que constituyen un tema no son exclusivas, por lo que se podrían encontrar palabras que se repiten dentro de temas distintos, quizás con matices diferentes.

En esta sección se procederá a generar modelos LDA con las siguientes variables parametrizadas:

- $Topic$, con 7 valores posibles: Cultura, Economía, Internacional, Nacional, Sociedad, Deporte y Tecnología.
- $numTopics$, con 3 valores posibles: 20, 30 y 40.
- $VocabSize$, con 4 valores posibles: 2000, 3000, 4000 y 5000.

Por consiguiente, trabajaremos a partir de ahora con 84 modelos de LDA que, como es de imaginar, no se pueden representar todos en este proyecto, por lo que se mostrará el resultado obtenido solamente de algunas simulaciones.

En la figura 5.1 se representa un modelo de estructura latente de *subtopics* para el tema Cultura. En esta gráfica podemos observar claramente los siguientes sub-temas inferidos a partir de las palabras individuadas por LDA en el texto:

- **"Gala de los premios Goya de 2014"**: Cine, Cultura, Películas, Ministro, Gala, Wert.
- **"Muerte de Paco de Lucía"** : Música, Paco, Lucía, Flamenco, Hermano.
- **"Libros y Poesía"**: Vida, Libros, Poeta, Poesía, Literatura, Palabras.
- **"Muerte del actor Philip Hoffman"**: Nueva, York, Muerte, Heroína, Policía.
- **"Facturación en taquillas"**: Millones, Euros, Salas, Media, 2012.
- **"Arte y museos"**: Arte, Obras, Artista, Museo, Nacional, Londres.

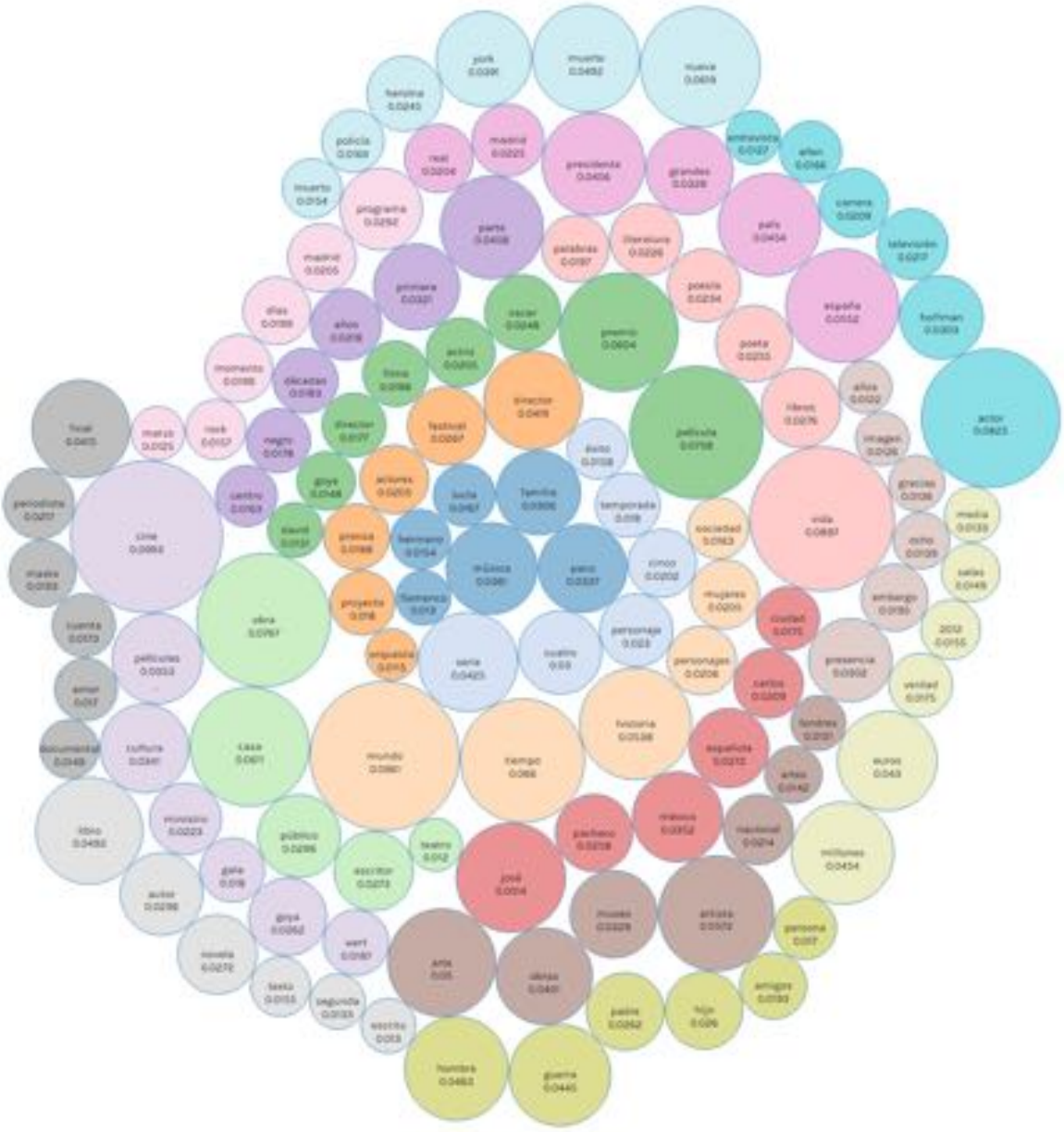


Figura 5.1: Representación clusterizada del modelo de Cultura para un diccionario de 2000 palabras y una estructura latente de 20 *subtopics*.

En el *Apéndice B* se incluyen las representaciones gráficas de más modelos, obtenidas tras las distintas simulaciones.

5.3 Comparación con ficheros no perteneciente al corpus

En esta sección se ha decidido implementar un sistema cuya entrada esté constituida por los modelos generados en la sección anterior y además, reciba como argumento noticias clasificadas no incluidas en el corpus de entrenamiento de los modelos. La salida de este sistema vendrá dada por la semejanza de dicho fichero de entrada con los distintos $subtopics$ de cada tema.

Para ello se ha empleado la función **topicDistributions()** que recibe como parámetro el documento a analizar y se aplicará sobre cada uno de los modelos generados en la sección anterior. Para proceder con esta implementación es necesario seguir los pasos de pre-procesado de Tokenización y CountVectorization descritos en el capítulo anterior. Para poder crear la matriz de ocurrencia del nuevo documento, es necesario modificar el código descrito en la anterior sección, para guardar el diccionario empleado en cada modelo ya que se necesitará para crear dicha matriz.

Dado que analizaremos todos los temas en conjunto, las demás variables parametrizadas conformarán 12 simulaciones independientes. El sistema devolverá, por lo tanto, por cada documento $numTopics$ valores, no ordenados, que representan la similitud del fichero externo al corpus con respecto a cada uno de los 7 modelos generados con esa determinada relación de $VocabSize$ y $numTopics$.

En la Figura 5.3 se muestra la salida del sistema para una noticia publicada en la sección de Cultura de El País de marzo de 2016, completamente desligada del corpus original que es de 2014 (Extracto de la misma en la figura 5.2). Los modelos empleados cuentan con un diccionario de 2000 palabras y una estructura latente de 20 $subtopics$.

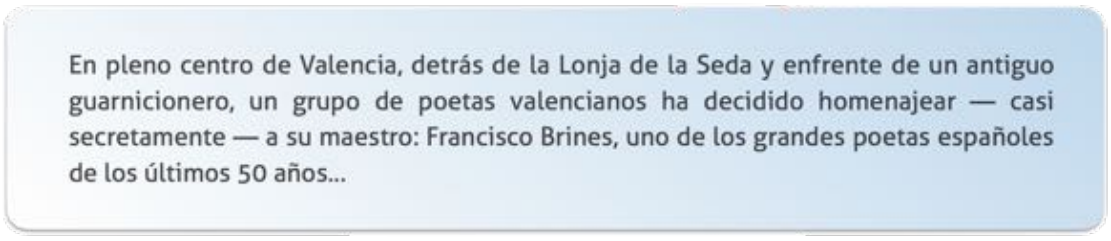

En pleno centro de Valencia, detrás de la Lonja de la Seda y enfrente de un antiguo guarnicionero, un grupo de poetas valencianos ha decidido homenajear — casi secretamente — a su maestro: Francisco Brines, uno de los grandes poetas españoles de los últimos 50 años...

Figura 5.2: Extracto del fichero de cultura analizado.

Como se puede observar en esta la figura 5.3, cada línea representa la similitud de este fichero con cada *subtopic* del tema en cuestión. En estos ejemplos la línea azul corresponde a los *subtopics* de Cultura, la negra a los de Economía, la verde a Internacional, la naranja a Nacional, la magenta a Deporte y la amarilla a Tecnología.

Figura 5.3: Representación de la salida del sistema para un fichero de Cultura de marzo de 2016, para un diccionario de 2000 palabras y una estructura latente de 20 *subtopics*.

Hay que precisar que esta función de DistributedLDAModel distribuye la probabilidad entre los *subtopics*, asignando a todos una probabilidad de semejanza y que la suma de dichas probabilidades es siempre igual a 1. Por lo tanto, podemos observar que el algoritmo, al distribuir la probabilidad entre todos los *subtopics*, para *numTopics* no muy elevados, hace que los valores de cada uno de ellos sean parecidos entre sí. Concretamente en la figura 5.3 se puede observar un pico elevado para el *subtopic* 8 del tema Cultura, así como otros valores de pico de diferentes temas.

Al aumentar el número de *subtopics*, la probabilidad media de la semejanza disminuye, haciendo que la probabilidad total se distribuya uniformemente entre más *subtopics* y afloren picos más elevados en temas más concretos.

Este fenómeno se aprecia claramente en las siguientes gráficas en las que se ha aumentado el número de *subtopics* a 30 y sucesivamente a 40 (véanse figuras 5.4 y 5.5).

Figura 5.4: Salida del sistema para el mismo fichero para un diccionario de 2000 palabras y una estructura latente de 30 *subtopics*.

Como si se tratara de una señal, conforme aumentamos el número de ***subtopics*** el ruido estadístico disminuye y se distribuye uniformemente entre los ***subtopics*** de otros temas, mientras que para el ***subtopic*** 4 de Cultura la probabilidad se hace cada vez más grande con respecto a los demás valores.

Figura 5.5: Salida del sistema para el mismo fichero para un diccionario de 2000 palabras y una estructura latente de 40 *subtopics*.

Por último, en la gráfica 5.5 vemos cómo el ***subtopic*** 10 de Cultura sigue teniendo una probabilidad elevada, por lo que podemos afirmar que se podría inferir cierta relación entre el documento analizado y el tema Cultura.

Este mismo análisis, se ha llevado al cabo sobre una noticia de economía (véase extracto en la figura 5.9) empleando un diccionario de 5000 palabras. Los resultados se pueden observar en las figuras 5.6, 5.7 y 5.8.

Figura 5.6: Representación de la salida del sistema para un fichero de Economía de marzo de 2016, para un diccionario de 5000 palabras y una estructura latente de 20 *subtopics*.

Figura 5.7: Salida del sistema para el mismo fichero de Economía, para un diccionario de 5000 palabras y una estructura latente de 30 *subtopics*.

Figura 5.8: Salida del sistema para el mismo fichero de Economía, para un diccionario de 5000 palabras y una estructura latente de 40 *subtopics*.

Acciona, FCC, Ferrovial y ACS lideran cuatro de los nueve consorcios que Reino Unido ha seleccionado para que compitan por obras de construcción de la línea ferroviaria de Alta Velocidad (AVE) que unirá Londres y Birmingham, un proyecto presupuestado en entre 7.100 y 11.800 millones de libras (entre unos 9.000 y 15.000 millones de euros)...

Figura 5.9: Extracto del fichero de economía analizado.

En las gráficas anteriores se ha podido analizar visualmente sólo el valor del primer $subtopic$ con mayor similitud con respecto al fichero analizado. Para un análisis más exhaustivo, se podría incluir la comparación de los n primeros $subtopics$ de cada tema con mayor similitud al documento bajo estudio. Para ello ha sido necesario modificar el código para que el valor de todos los $subtopics$ esté ordenado de mayor a menor y se guarde la suma de los n primeros.

En el siguiente ejemplo, se ha tomado un fichero procedente de la sección Internacional de El País de marzo de 2016 (extracto en la figura 5.13). Como se puede ver en la figura 5.10, para un diccionario de 5000 palabras y una estructura latente de 40 $subtopics$, el anterior análisis arroja un resultado erróneo, para un sistema que se limita a escoger el primer pico de similitud. De hecho, hay un claro pico en el $subtopic$ 39 del tema de Sociedad, cuyo valor es del 19% de similitud con respecto al fichero. Sin embargo, podemos observar también que hay otros dos valores altos: un pico en el $subtopic$ 8 del tema Internacional, cuyo valor es 15,25% y otro pico en el $subtopic$ 20 también del tema Internacional, cuyo valor es 14,05%.

Figura 5.10: Representación de la salida del sistema para un fichero en la categoría Internacional de Marzo de 2016, para un diccionario de 5000 palabras y una estructura latente de 40 $subtopics$.

A continuación realizamos el nuevo análisis de los n-$subtopics$ más comunes y obtendremos los resultados representados en la figura 5.11, en los que se aprecia cómo la suma de los n $subtopics$ determina que para $n > 2$ el sistema sería capaz de detectar el tema del fichero.

Figura 5.11: Representación de la suma de los n-$subtopics$ más frecuentes por cada tema, para un fichero en la categoría Internacional de marzo de 2016, con un diccionario de 5000 palabras y una estructura latente de 40 $subtopics$.

Esto se debe a que un fichero, sobre todo si es muy grande, podría tener coincidencias con más de un $subtopic$ dentro de cada modelo, por lo que la probabilidad conjunta de pertenecer a un tema en concreto, se distribuiría entre los distintos $subtopics$ del mismo. En la figura 5.12 se han analizado los mismos resultados empleando una gráfica Kiviat en la que podemos apreciar claramente la tendencia de la suma de los n-$subtopics$. Para $n = 1$ tenemos dos temas preponderantes: Sociedad e Internacional. Conforme aumenta n Sociedad aumenta linealmente por un factor de $1/numTopics$ mientras que Internacional dobla su valor para $n = 2$.

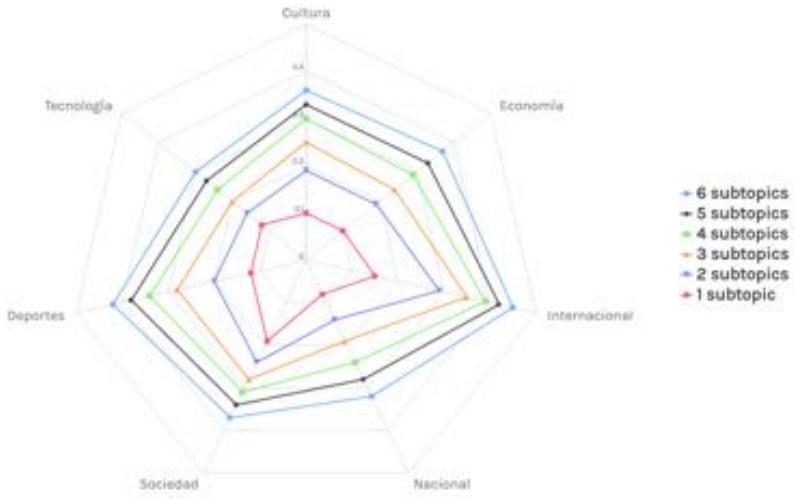


Figura 5.12: Representación de un diagrama de Kiviat con la suma de los n-$subtopics$ más frecuentes por cada tema, para un fichero de la categoría Internacional de Marzo de 2016, con un diccionario de 5000 palabras y una estructura latente de 40 $subtopics$.

El Tribunal Supremo de Brasil se reserva las investigaciones sobre Lula. La decisión mantiene fuera del caso al juez Sérgio Moro, que investiga las corruptelas en Petrobras. El Supremo Tribunal Federal (STF) brasileño ha decidido este jueves quedarse por ahora con la investigación sobre el expresidente Luiz Inácio Lula da Silva...

Figura 5.13: Extracto del fichero Internacional analizado.

Para terminar esta sección, se puede concluir que el tema que ha demostrado arrojar mejores resultados ha sido Deportes, probablemente debido a que la estructura de las noticias de este tipo de tema es bastante parecida y repetitiva. Además el vocabulario, presumiblemente, es bastante ortogonal a los diccionarios empleados para los demás temas. Véase el ejemplo de una noticia de baloncesto (extracto en figura 5.17) y los resultados para un tamaño de diccionario de 5000 palabras y para 20, 30 y 40 *subtopics* (figuras 5.14, 5.15 y 5.16)

Figura 5.14: Representación de la salida del sistema para un fichero de Deportes de Marzo de 2016, para un diccionario de 5000 palabras y una estructura latente de 20 *subtopics*.

Figura 5.15: Salida del sistema para el mismo fichero de Deportes, para un diccionario de 5000 palabras y una estructura latente de 30 *subtopics*.

Figura 5.16: Salida del sistema para el mismo fichero de Deportes, para un diccionario de 5000 palabras y una estructura latente de 40 $subtopics$.

Un Mirotic superlativo da nueva vida a los Bulls. Chicago gana a Indiana gracias a una canasta de Butler y a los 28 puntos del hispano-montenegrino. Un recital de Nikola Mirotic y una canasta de Jimmy Butler a tres segundos para el final en la cancha de Indiana (96-98) le dieron a Chicago un triunfo que mantiene sus posibilidades de clasificación para los 'playoffs'...

Figura 5.17: Extracto del fichero de Deportes analizado.

5.4 Optimización del sistema

Una vez obtenidos los resultados de los apartados anteriores, es necesario poder extender el análisis y generalizar el sistema para que se pueda optimizar y obtener los valores de $VocabSize$ y $numTopics$ que hagan máxima la tasa de acierto. Para ello, se procede a hacer una validación cruzada, en la que cada corpus de cada tema se dividirá en 4 partes.

Se procederá a generar modelos con la combinación de tres sub-partes para luego utilizar como entrada del sistema la cuarta fracción no empleada para la generación del modelo. Este proceso se realizará para todas las posibles combinaciones y todos los temas.

En la figura 5.18, se puede ver un esquema de la validación cruzada que se ha empleado en esta sección. En ésta se puede observar como se emplea la fracción primera, segunda y tercera del tema Cultura, obteniendo un corpus de entrenamiento (**CT4_123**). Este corpus se empleará para la generación de los modelos LDA y se usará la parte cuarta, excluida del corpus de Cultura, para el

test. Este proceso se reitera para las restantes partes, generando los modelos **CT4_234**, **CT4_341**, **CT4_412**. El mismo trabajo se realizará para todos los temas presentes en el corpus.

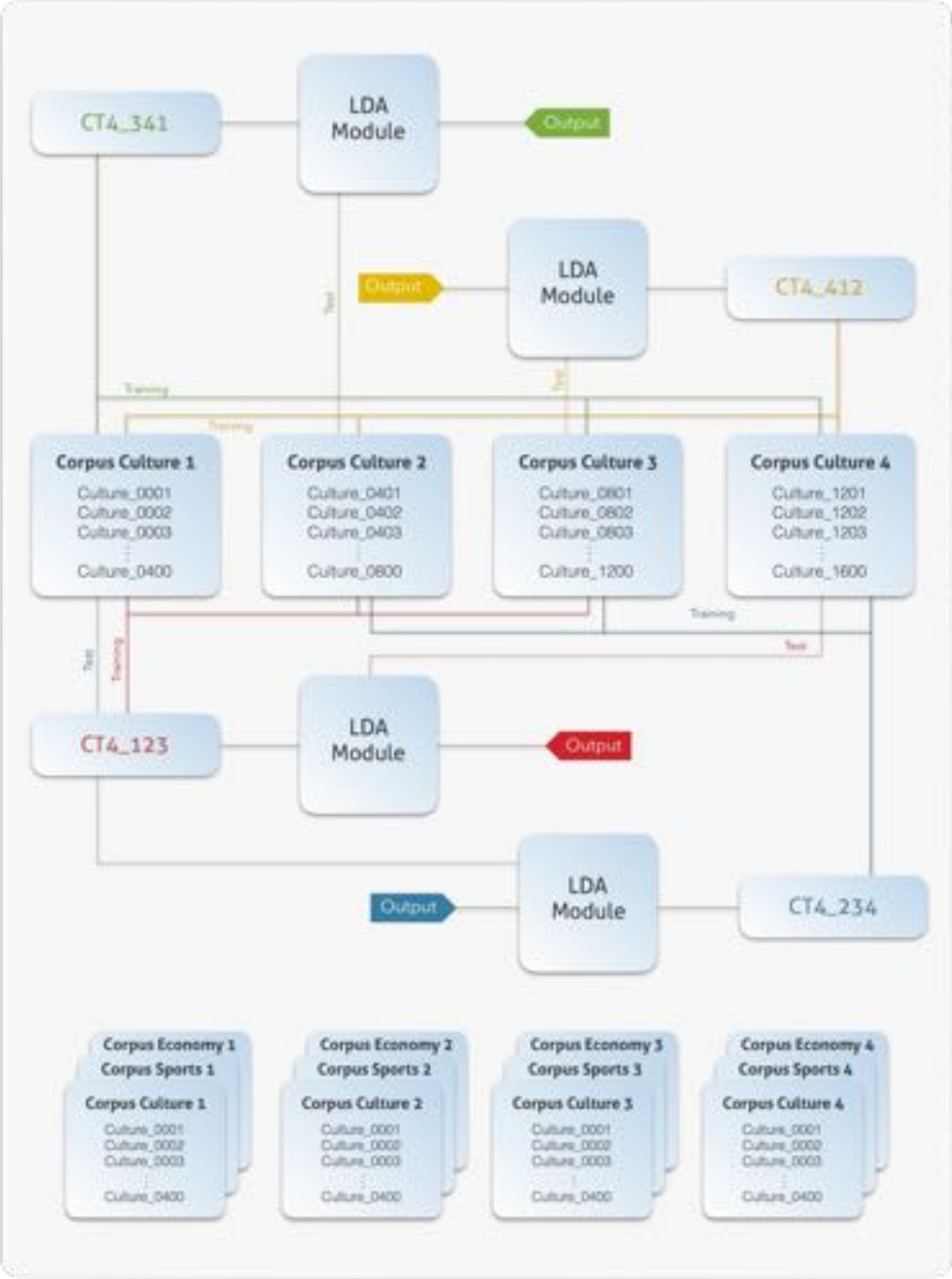


Figura 5.18: Esquema de validación cruzada.

En un primer momento se recogerán todos los resultados de cada tema en una misma gráfica para poder observar la distribución que toman los datos. Como podemos observar en la figura 5.19, los valores medios de semejanza con respecto al fichero crecen conforme aumentamos el número de palabras del diccionario y, por el contrario, disminuyen conforme aumentamos el número de *subtopics*, tal como se ha descrito en la sección anterior. Se muestra sólo una de las gráficas porque la tendencia en todos los temas es bastante parecida.

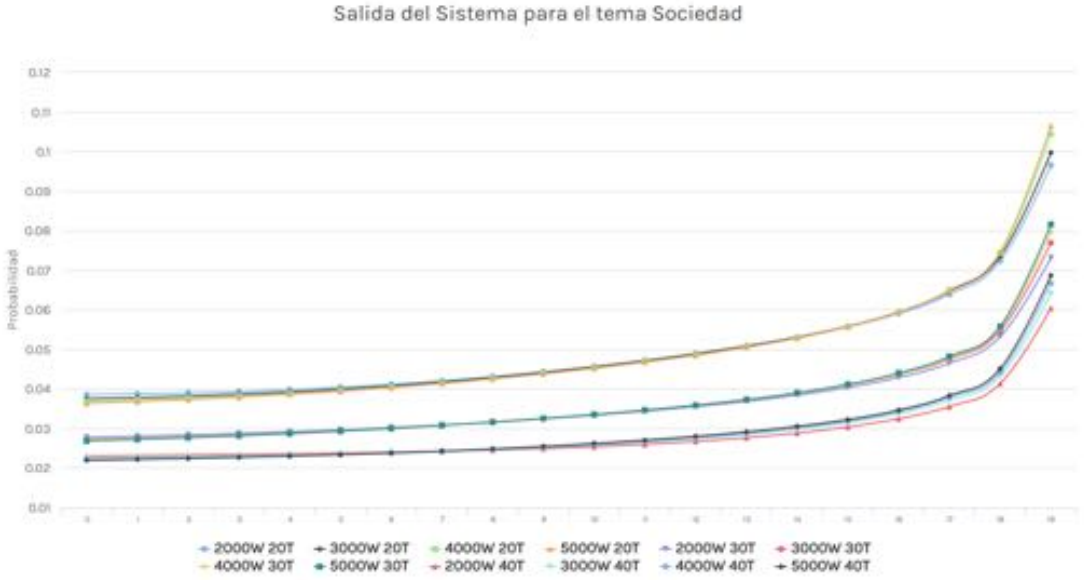


Figura 5.19: Resultados de los valores medios del tema Sociedad.

En la figura 5.20 vamos a analizar los mismos datos teniendo en cuenta, además de la media, la desviación y representaremos las gaussianas resultantes.

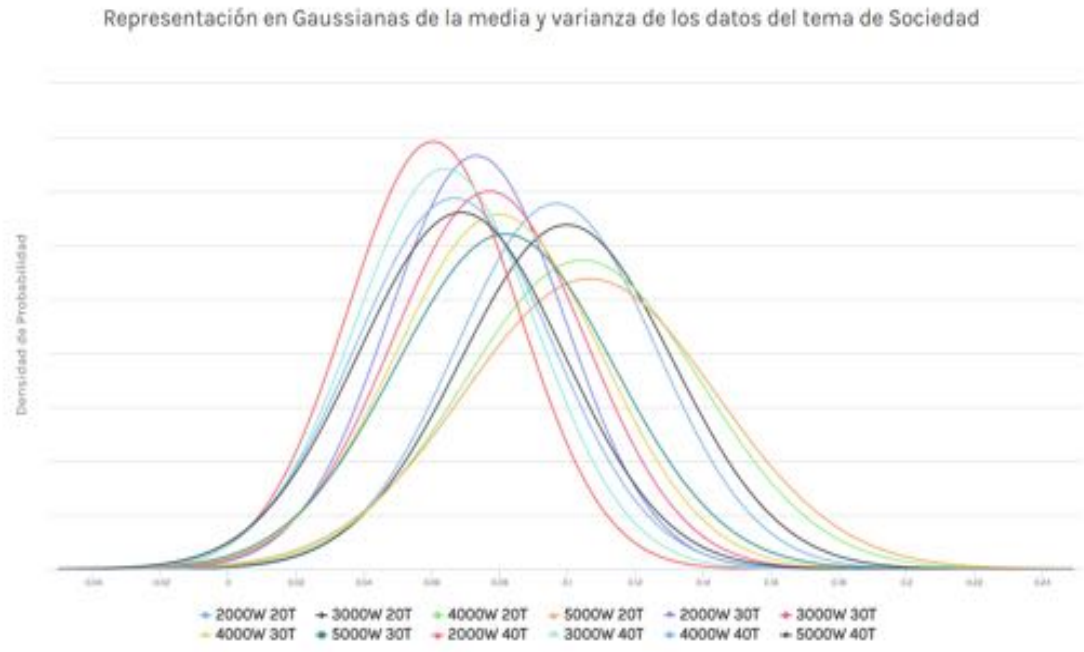


Figura 5.20: Representación de los valores medios y las desviaciones de los valores obtenidos para el tema Sociedad.

Como se puede observar en la figura 5.20, las gaussianas nos muestran el fenómeno de desplazamiento de la media a la izquierda en base al aumento de los $subtopics$ y el desplazamiento a la derecha conforme aumentamos el tamaño del diccionario, como hemos podido apreciar en la gráfica anterior. Además, nos añaden otro dato interesante: conforme aumentamos el número de $subtopics$ y disminuimos el número de palabras, las gaussianas se hacen más estrechas. Desviaciones más pequeñas de la distribución de los valores nos indican que el conjunto de datos es más homogéneo y más predecible para ese tema en concreto.

A partir de estas premisas, podremos proceder con la validación cruzada propiamente dicha, en la que cada fichero se analizará con los 7 modelos de cada tema en los que dicho fichero no está incluido. En este análisis se tendrá en cuenta sólo el valor del primer pico y se elegirá el tema cuyo valor sea máximo.

En el análisis se mostrarán las tasas de acierto para cada una de las 12 simulaciones sobre cada tema, obtenidas al analizar los 400 ficheros más grandes de cada sub-grupo de validación cruzada (1600 por cada tema). Esta restricción se ha aplicado porque los tiempos de simulación resultaban muy grandes para los temas Internacional y Nacional. El número de ficheros se ha escogido a partir del tema con un menor número de archivos que es Cultura.

A continuación se muestran las tablas con las tasas de acierto de cada una de las 12 simulaciones para todos los temas analizados (Tablas 5.1 – 5.7).

Como se puede observar en dichas tablas, la variación de los parámetros $VocabSize$ y $numTopics$ es crucial a la hora de generar modelos ya que, en media, supone un incremento de la tasa de acierto en más de 10 puntos porcentuales, pudiendo alcanzar 20 puntos porcentuales para el tema de Cultura.

Tasa de Aciertos del tema Cultura

Vocabsize		2000	3000	4000	5000
numTopics	**20**	32,23 %	26,20 %	26,45 %	26,70 %
	30	42,86 %	37,27 %	35,26 %	34,94 %
	40	36,29 %	**47,21 %**	43,58 %	41,28 %

Tabla 5.1: Tasa de aciertos de Cultura para las distintas simulaciones.

Tasa de Aciertos del tema de Economía

Vocabsize		2000	3000	4000	5000
numTopics	**20**	31,9 %	34,60 %	36,02 %	**39,80 %**
	30	32,15 %	32,66 %	34,34 %	33,92 %
	40	28,90 %	30,56 %	27,27 %	31,23 %

Tabla 5.2: Tasa de aciertos de Economía para las distintas simulaciones.

Tasa de Aciertos del tema Internacional

Vocabsize		2000	3000	4000	5000
numTopics	**20**	42,82 %	39,14 %	40,91 %	**43,69 %**
	30	42,57 %	40,00 %	39,37 %	43,61 %
	40	41,81 %	33,83 %	35,77 %	38,69 %

Tabla 5.3: Tasa de aciertos de Internacional para las distintas simulaciones.

Tasa de Aciertos del tema Nacional

Vocabsize		2000	3000	4000	5000
numTopics	**20**	38,89 %	35,01 %	33,17 %	33,17 %
	30	36,67 %	36,02 %	34,51 %	35,61 %
	40	**39,59 %**	30,53 %	36,09 %	34,59 %

Tabla 5.4: Tasa de aciertos de Nacional para las distintas simulaciones.

Tasa de Aciertos del tema Sociedad

Vocabsize		2000	3000	4000	5000
numTopics	**20**	44,42 %	47,97 %	47,47 %	49,49 %
	30	49,11 %	49,82 %	**53,55 %**	50,13 %
	40	50,51 %	48,36 %	52,02 %	52,14 %

Tabla 5.5: Tasa de aciertos de Sociedad para las distintas simulaciones.

Tasa de Aciertos del tema Deportes

Vocabsize		2000	3000	4000	5000
numTopics	**20**	54,68 %	62,15	60,35 %	61,87 %
	30	63,64 %	62,03 %	62,09 %	64,56 %
	40	59,28 %	57,91 %	**65,57 %**	60,46 %

Tabla 5.6: Tasa de aciertos de Deportes para las distintas simulaciones.

Tasa de Aciertos del tema Tecnología

Vocabsize		2000	3000	4000	5000
numTopics	**20**	**23,14 %**	21,01 %	19,34 %	13,99 %
	30	15,48 %	16,12 %	18,27 %	18,43 %
	40	14,47 %	14,47 %	14,91 %	10,23 %

Tabla 5.7: Tasa de aciertos de Tecnología para las distintas simulaciones.

En la siguiente sección analizaremos en detalle los datos obtenidos tras la validación cruzada.

5.5 Análisis de resultados

En esta sección analizaremos los resultados que la validación cruzada ha puesto de manifiesto, empezando por considerar las tasas de aciertos máximas por cada uno de los temas.

Como se puede apreciar en la tabla 5.8, las tasas máximas de acierto se sitúan, en la mayoría de los temas, alrededor del 40% teniendo un repunte del 65,57% para el tema de Deportes.

Tasas de Acierto

	Cultura	Economía	Internacional	Nacional	Sociedad	Deportes	Tecnología
Tasa de Acierto	47,21 %	39,80 %	43,69 %	39,59 %	53,55 %	**65,57 %**	23,14 %
VocabSize	3000	5000	5000	2000	4000	**4000**	2000
numTopics	40	20	20	40	30	**40**	20

Tabla 5.8: Valor de la Tasa de Acierto en función de los temas y de los parámetros óptimos.

A continuación, en las figuras 5.21 – 5.28 se mostrará la salida del sistema para los valores de $VocabSize$ y $numTopics$ que han maximizado la tasa de acierto en cada tema.

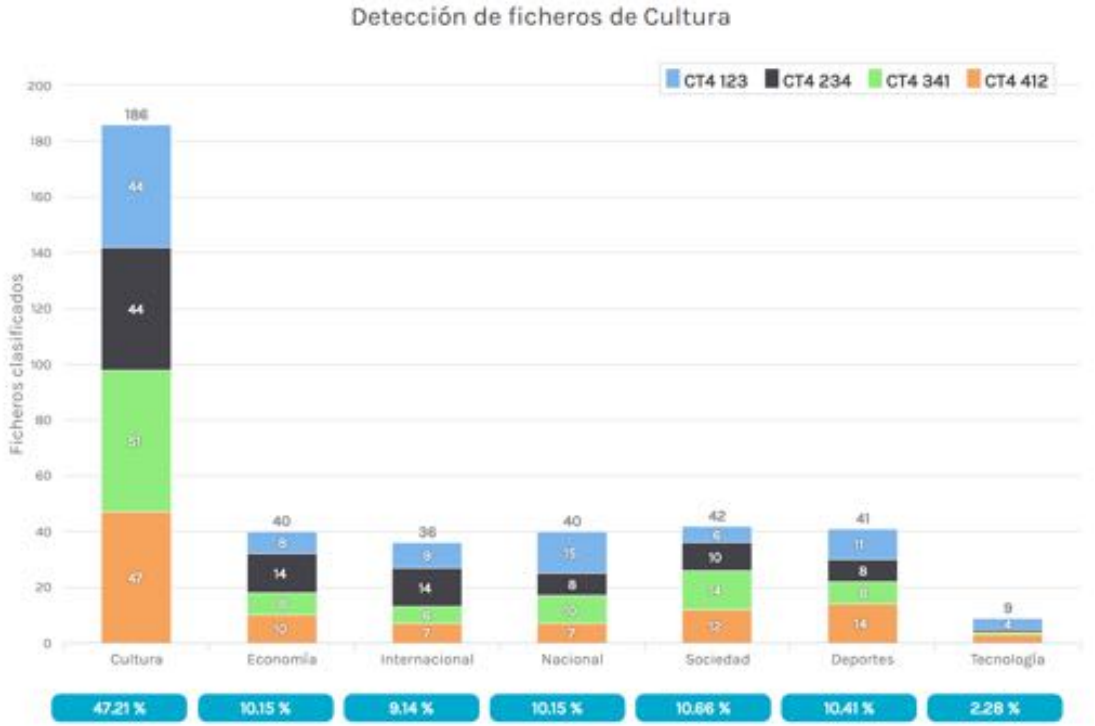


Figura 5.21: Salida del sistema para la sub-sección de Cultura para tamaño de diccionario de 3000 palabras y estructura latente de 40 $subtopics$.

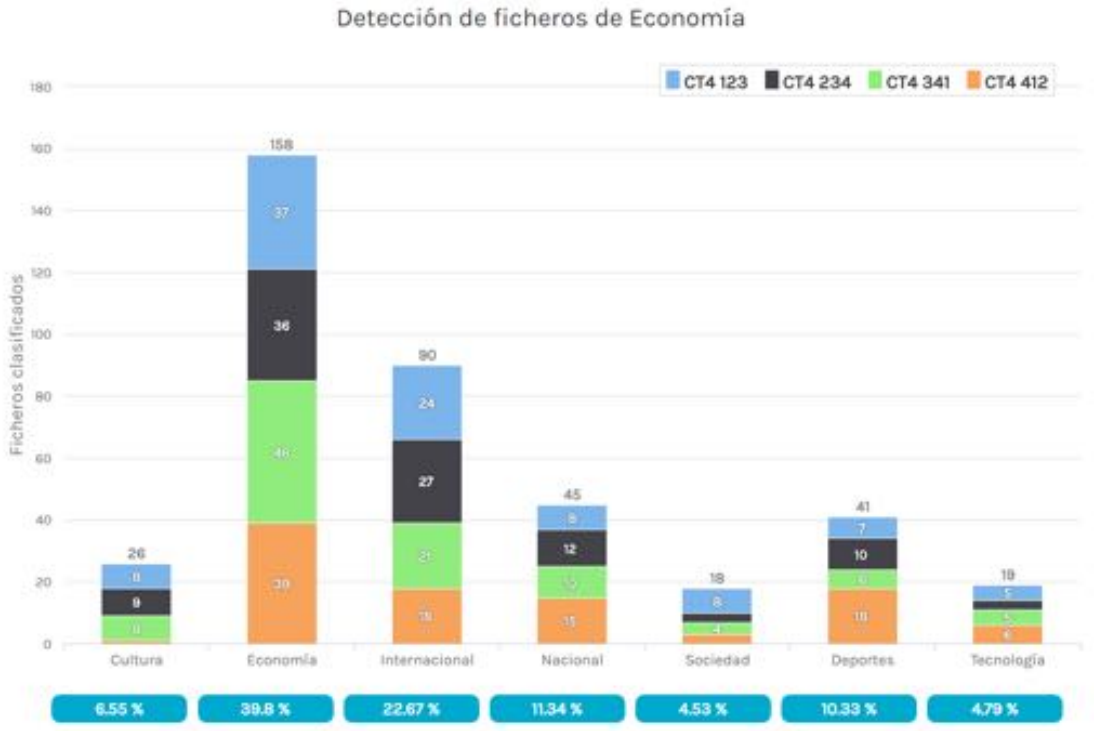


Figura 5.22: Salida del sistema para la sub-sección de Economía para tamaño de diccionario de 5000 palabras y estructura latente de 20 $subtopics$.

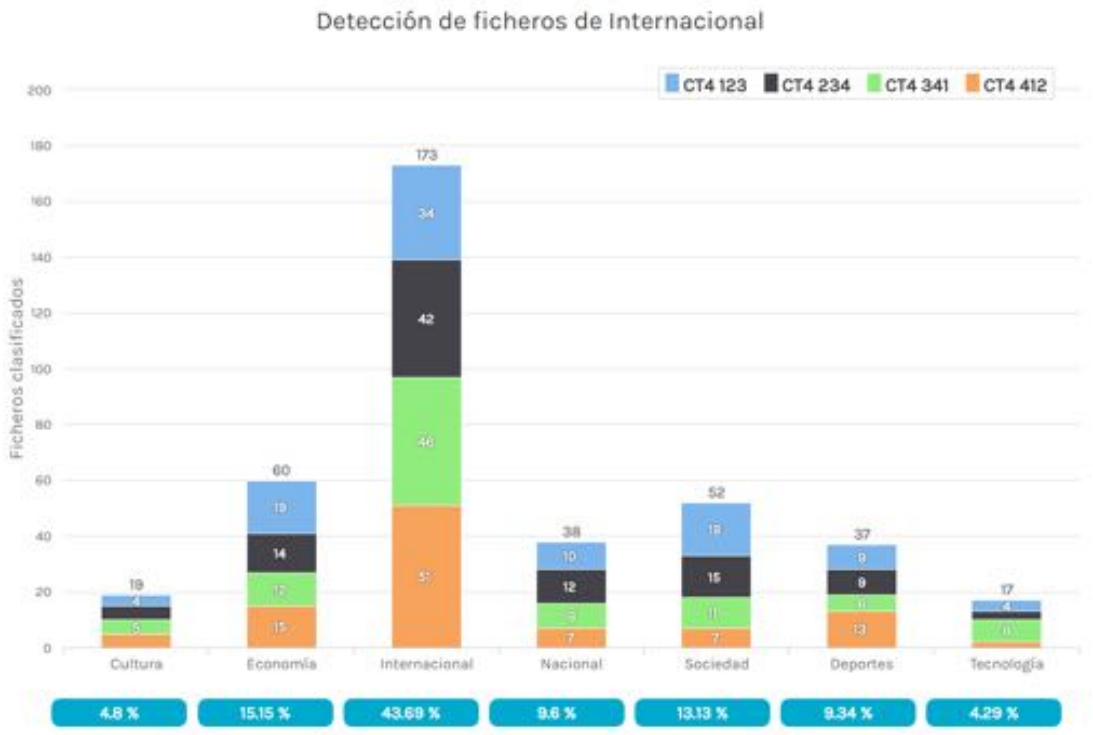


Figura 5.23: Salida del sistema para la sub-sección de Internacional para tamaño de diccionario de 5000 palabras y estructura latente de 20 *subtopics*.

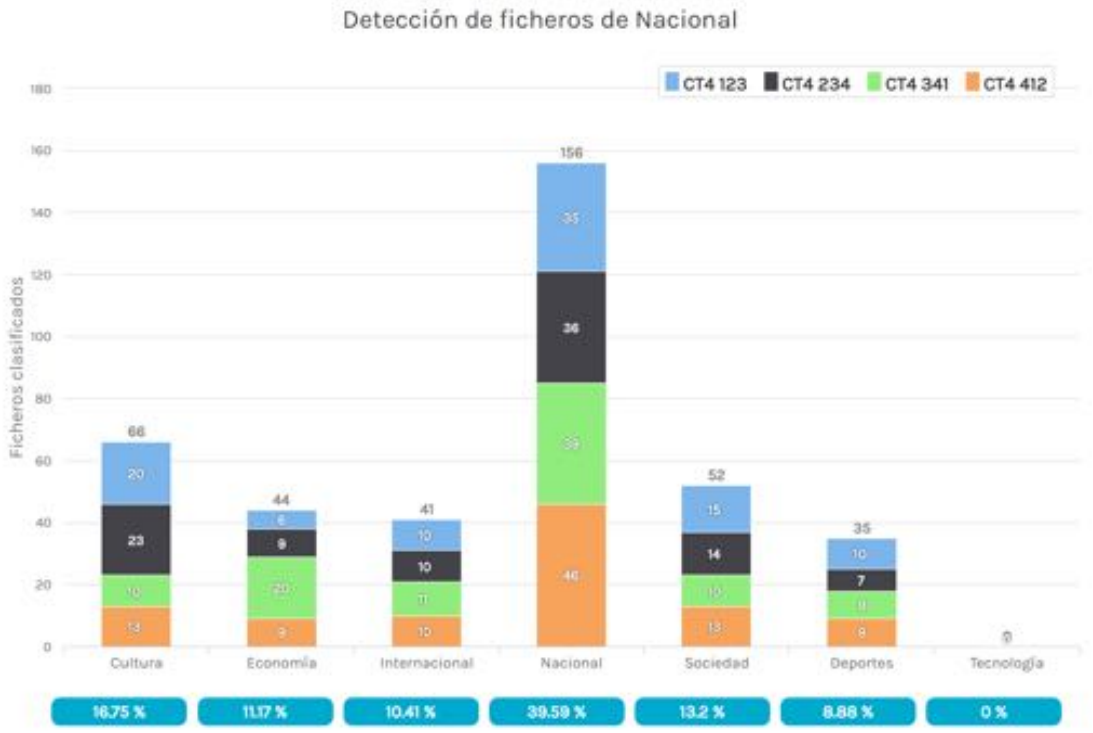


Figura 5.24: Salida del sistema para la sub-sección de Nacional para tamaño de diccionario de 2000 palabras y estructura latente de 40 *subtopics*.

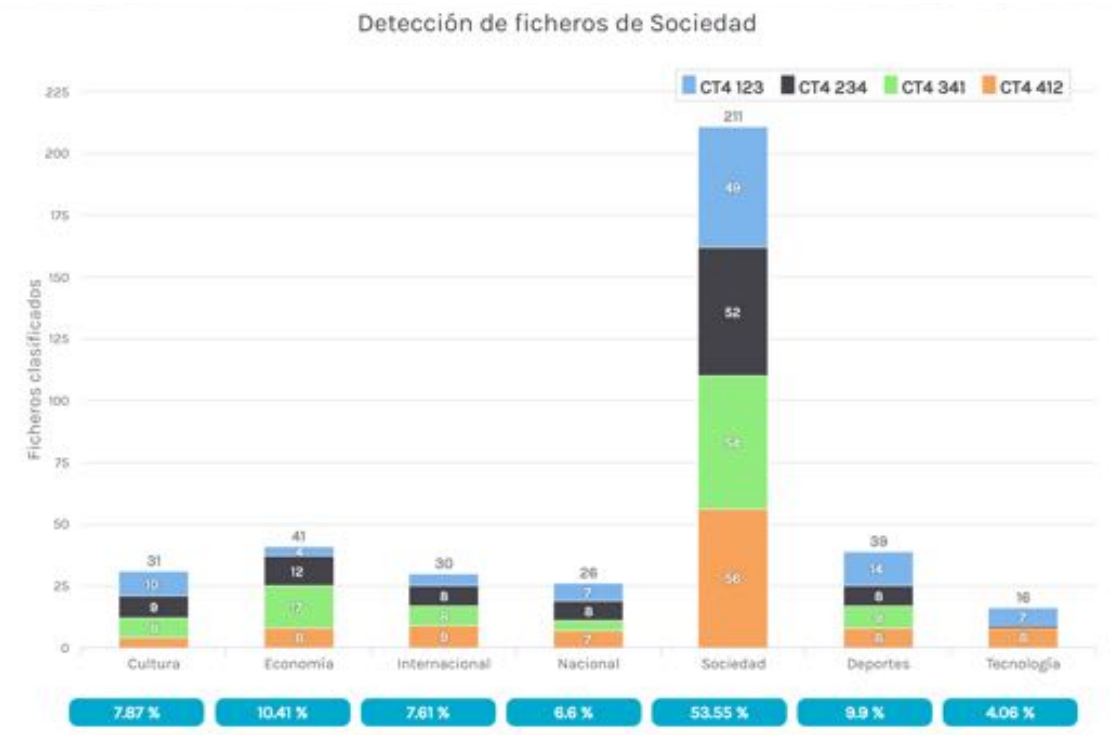


Figura 5.25: Salida del sistema para la sub-sección de Sociedad para tamaño de diccionario de 4000 palabras y estructura latente de 30 *subtopics*.

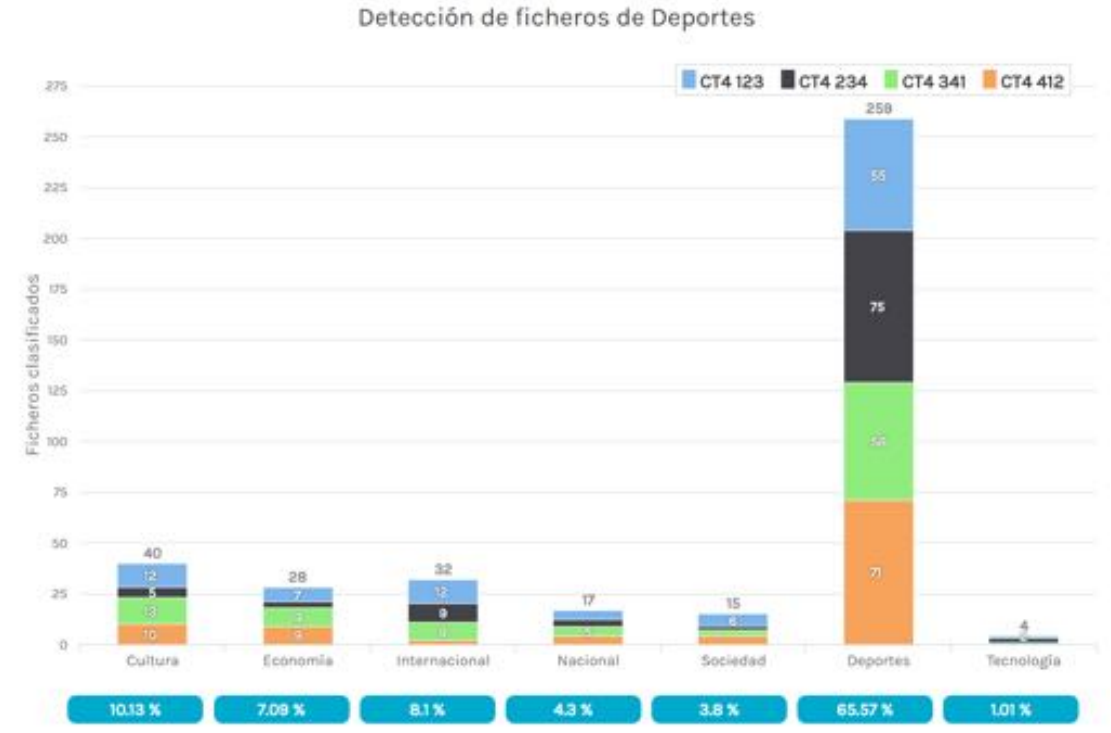


Figura 5.26: Salida del sistema para la sub-sección de Internacional para tamaño de diccionario de 4000 palabras y estructura latente de 40 *subtopics*.

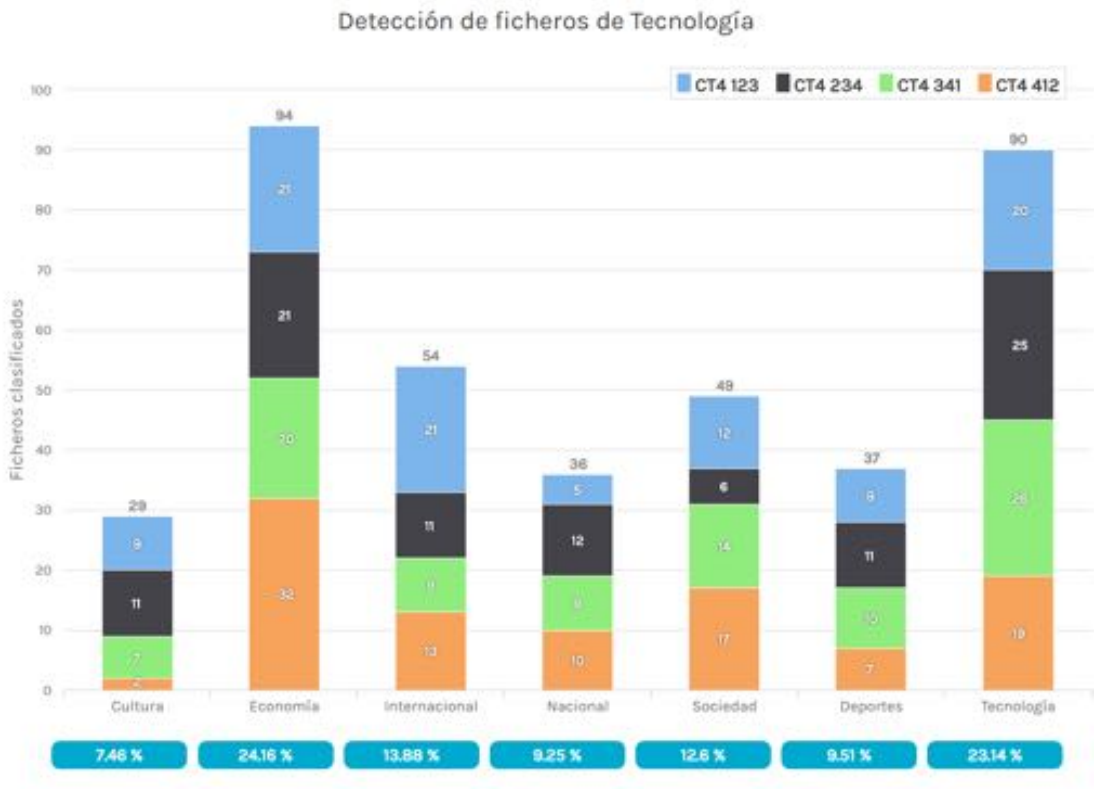


Figura 5.27: Salida del sistema para la sub-sección de Tecnología para tamaño de diccionario de 2000 palabras y estructura latente de 20 *subtopics*.

Las altas tasas de acierto alcanzadas en las pruebas del tema Deportes ponen de manifiesto que éste tiene una estructura temática bastante característica y que se repite a lo largo de la colección de los documentos. Dicha característica ya se había podido apreciar en las anteriores pruebas realizadas en el capítulos 4.

Por el contrario, para el tema Tecnología las tasas de acierto son muy bajas, siendo el valor máximo alcanzado del 23,14% para 2000 palabras y 20 *subtopics*. Probablemente esto se debe al vocabulario específico y reducido de este tema así como a la presencia de lenguaje de esta sección. Esta característica del tema de Tecnología queda reflejada en la salida del sistema en la figura 5.27, ya que el sistema detecta en el 23,5% de los ficheros la presencia del tema de Economía. Además este fenómeno queda patente en el mismo modelo LDA generado previamente, ya que aparecen sub-temas relacionados con Economía, como se puede apreciar en al figura 5.28.

También resulta de interés notar que los ficheros de Economía son detectados con mucha facilidad con el tema Internacional y viceversa, probablemente por el lenguaje que emplean y por la presencia de noticias económicas en el ámbito internacional.

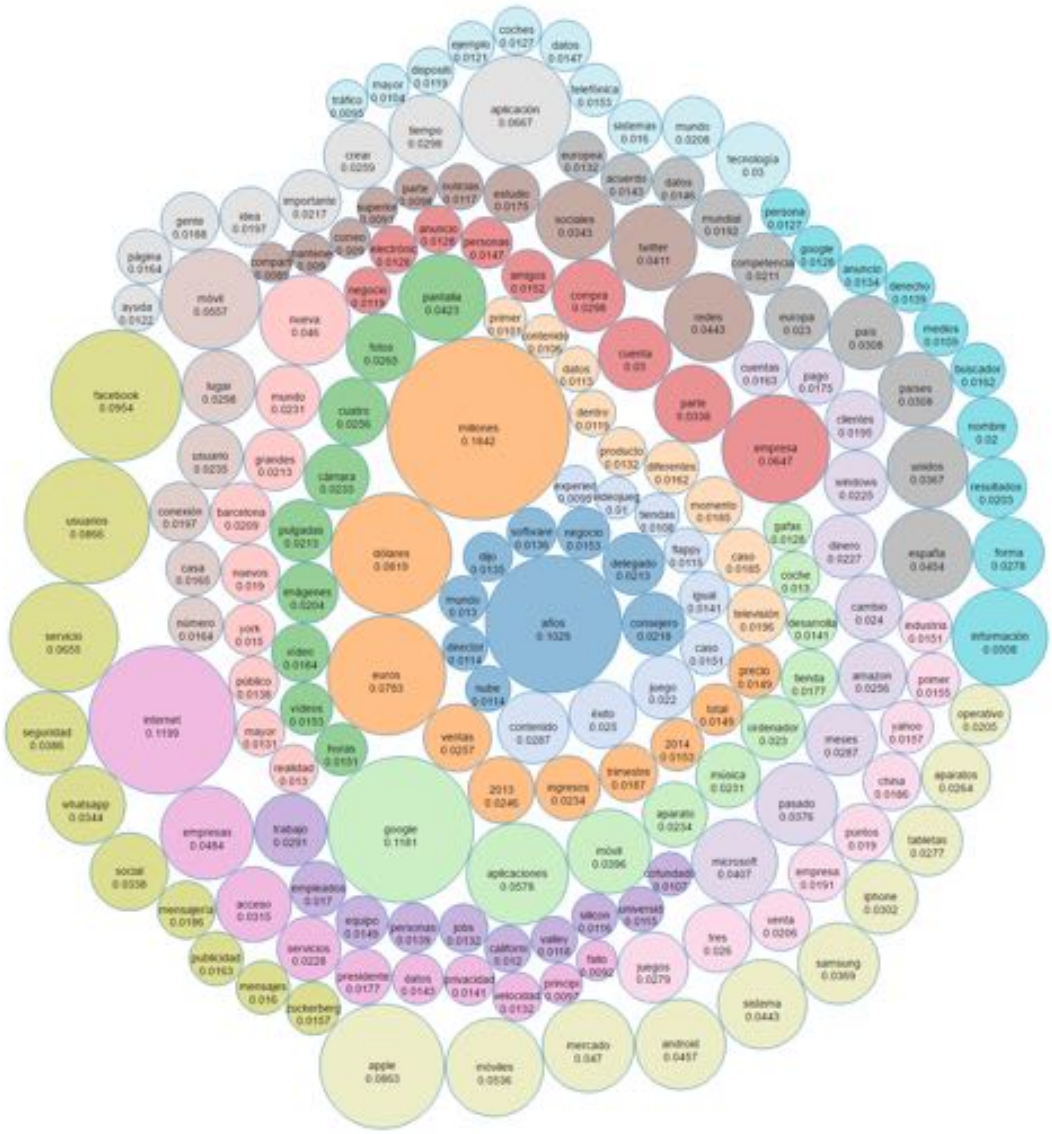


Figura 5.28: Gráfica clusterizada de un modelo LDA para un diccionario de 3000 palabras y una estructura latente de 20 *subtopics* para el tema de Tecnología.

A la hora de evaluar los resultados arriba mencionados se deberá también tener en cuenta que la clasificación previa de los textos presentes en el corpus no siempre es precisa, ya que una noticia puede clasificarse de forma diferente dependiendo de muchos factores. Así mismo en el periódico El País, una misma noticia puede aparecer en dos o más secciones.

En concreto, al analizar una noticia [51] de marzo de 2016 publicada en la sección de actualidad política de España de este periódico, el sistema proporciona una salida que en principio podría parecer errónea.

En la figura 5.29 se puede observar la salida del sistema para dicha noticia y en la figura 5.30 se puede observar el diagrama de Kiviat para la suma de los

$n\text{-}subtopics$ de cada tema. En ambas figuras se aprecia claramente cómo el sistema detecta una clara tendencia del documento a parecerse a una estructura temática de Sociedad. En concreto, en la figura 5.29 se puede observar un pico en el $subtopic$ 13 del tema Sociedad, mientras que el tema Nacional no parece estar relacionado con dicha noticia.

Figura 5.29: Salida del sistema para la noticia del País de la sección de actualidad de política de España.

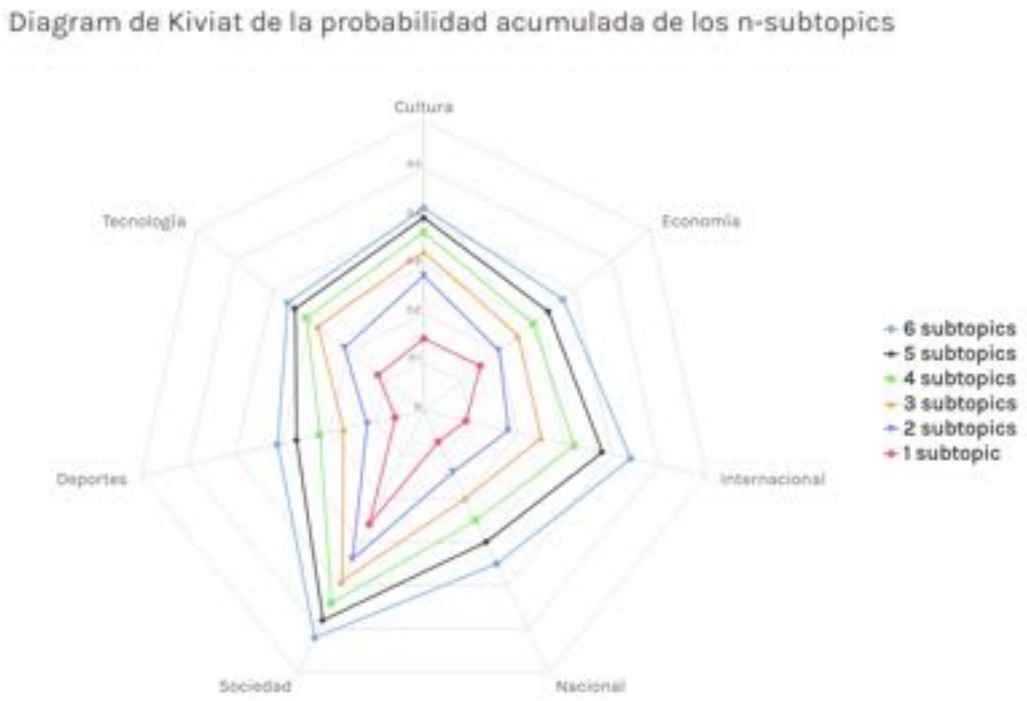
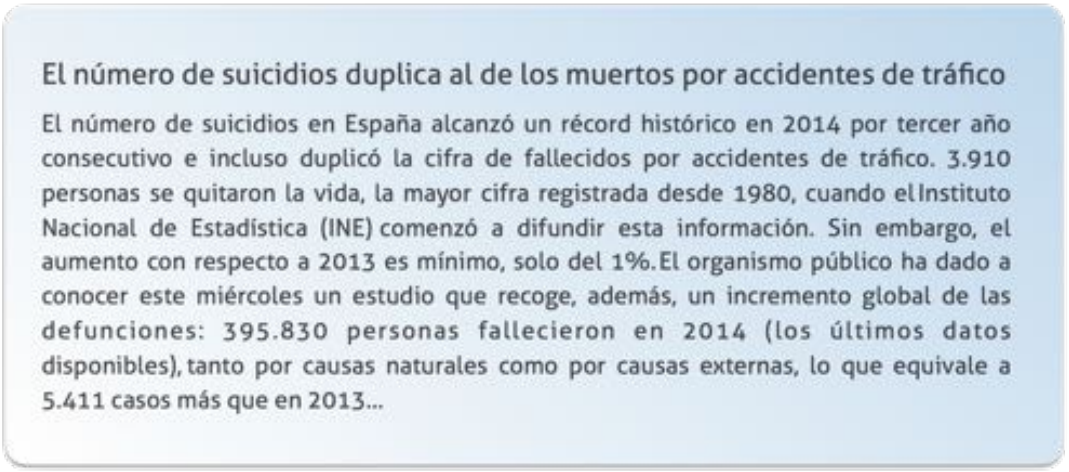


Figura 5.30: Diagrama de Kiviat de los $n\text{-}subtopics$ para el mismo fichero.

Al analizar manualmente el contenido del documento, véase figura 5.31, se puede observar que se trata de una noticia que habla de la tasa de suicidio en España, por lo que podemos concluir que el sistema ha acertado correctamente el tema del fichero, aunque no coincida con la categoría en la que se había clasificado.



El número de suicidios duplica al de los muertos por accidentes de tráfico

El número de suicidios en España alcanzó un récord histórico en 2014 por tercer año consecutivo e incluso duplicó la cifra de fallecidos por accidentes de tráfico. 3.910 personas se quitaron la vida, la mayor cifra registrada desde 1980, cuando el Instituto Nacional de Estadística (INE) comenzó a difundir esta información. Sin embargo, el aumento con respecto a 2013 es mínimo, solo del 1%. El organismo público ha dado a conocer este miércoles un estudio que recoge, además, un incremento global de las defunciones: 395.830 personas fallecieron en 2014 (los últimos datos disponibles), tanto por causas naturales como por causas externas, lo que equivale a 5.411 casos más que en 2013...

Figura 5.31: Extracto de la noticia analizada.

En el siguiente capítulo se exponen las conclusiones al presente Proyecto de Fin de Carrera y se delinea el trabajo futuro que se puede realizar a partir del mismo.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

Uno de los objetivos de este proyecto era implementar un sistema de inferencia de estructuras latentes de temas, a través del algoritmo Latent Dirichlet Allocation en grandes colecciones de textos. Para conseguirlo, se han realizado una serie de pruebas sobre herramientas y frameworks distintos, habiendo elegido posteriormente el que mejor se adapta a las exigencias y los objetivos del proyecto.

Se ha podido determinar que la mejor aplicación para dicho propósito es Apache Spark, ya que permite trabajar en un entorno de programación más amigable y, a su vez, ejecutar algoritmos iterativos de forma más rápida con respecto a otros frameworks distribuidos. Además, incluye una librería muy potente para el desarrollo de código orientado a Machine Learning.

Tras haber implementado el sistema de generación de modelos por cada una de las categorías en las que el corpus había sido clasificado manualmente con anterioridad, se ha creado una estructura temática. Así mismo, se han parametrizado dos variables muy importantes para la generación de los modelos: el tamaño del diccionario de cada tema, $VocabSize$, y el número de sub-estructuras que el algoritmo deberá encontrar en cada tema, $numTopics$.

El objetivo final ha sido evaluar la relación que existe entre el tema sobre el que tratan ficheros externos al corpus de entrenamiento y la similitud con los subtemas de las distintas estructuras temáticas de dicho corpus, pudiendo en definitiva elegir entre ellas la que tiene mayor semejanza con respecto a la entrada. Los

experimentos que se han realizado en las secciones anteriores han demostrado que existe cierto grado de relación entre noticias de prensa desligadas del corpus y las estructuras generadas previamente.

En una última fase del proyecto, se ha procedido a realizar un proceso de validación cruzada con el que se han optimizado las variables parametrizadas del sistema y se han encontrado los valores óptimos que hacen que se maximice la tasa de acierto de cada tema.

Como era de esperar, las tasas de acierto más elevadas de cada tema, se han obtenido para distintos valores de los dos parámetros que se han hecho variar a lo largo de las pruebas y oscilan, entre un 65,57% para Deportes y un 21,14% para Tecnología.

6.2 Trabajo futuro

Las tasas de aciertos de algunos de los temas mencionadas en el capítulo anterior son muy alentadoras, sobre todo para los temas de Deportes, Cultura y Sociedad ya que permiten afirmar que, con pequeñas mejores del sistema, dichas tasas de acierto podrían aumentar con facilidad.

El trabajo futuro podría estar orientado a mejorar dos aspectos del presente proyecto: las mejoras en el pre-procesamiento del corpus y la parametrización de variables adicionales del sistema de generación.

En cuanto al pre-procesamiento del corpus, habría que destacar la posibilidad de introducir en los textos el concepto de colocación. Las colocaciones son palabras que aparecen con frecuencia combinadas entre sí para formar nuevas unidades con significado léxico diferente al original de cada una de ellas. Algunos ejemplos podrían ser: *El Presidente de los Estados Unidos, Producto Interior Bruto, El campeón de la Champions, Real Madrid, iPhone 6 SE, El teatro Real,* etc.

Actualmente, para un texto de entrada como el que se puede apreciar en la Figura 6.1 se contaría la frecuencia de aparición de la palabra “Paco” como dos entidades iguales que se repiten 2 veces a lo largo del texto y lo mismo ocurriría para la palabra “León” cuyo número de apariciones también sería igual a 2 (recordamos que el sistema ignora las mayúsculas). Hacer el sistema sensible a las mayúsculas no resolvería el problema ya que la palabra “Paco” seguiría tratándose como una misma entidad.

Figura 6.1: Ejemplo de corpus.

Creando una lista de colocaciones, detectaríamos "*Paco_de_Lucía*" como un token independiente de "*Paco_León*" y también seríamos capaces de diferenciarlo del nombre común "*león*". Probablemente con este pre-procesado del corpus se mejorarían todos los modelos, ya que en la prensa abundan todo tipo de colocaciones y frases hechas. Inclusive, habría que estudiar la utilidad de crear una lista de colocaciones de stopwords para eliminarlas también del corpus antes de generar los modelos.

En cuanto a la parametrización del sistema generador de los modelos, cabría decir que hay dos valores que no han sido parametrizados en este proyecto y que son, desde el principio, valores fijos. En concreto, se podrían hacer unos experimentos variando el valor de $minDF$, el número mínimo de documentos en los que una palabra tiene que aparecer para ser contada y $minTF$, el número mínimo de apariciones de la palabra en un mismo documento para ser contada.

Por último, pero no por ello menos interesante, se podría evaluar la factibilidad de crear un sistema capaz de aprender a partir de los ficheros externos al corpus. En caso de fallo a la hora de clasificar el fichero, el sistema podría tomar la decisión de incluir el texto en la base de datos para re-generar el modelo que ha fallado para dicha clasificación. Este proceso debería, además, ser complementado con una actualización constante de la lista de colocaciones y de stopwords.

Bibliografía:

Capítulo 1

[1] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." University of Massachusetts. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://mallet.cs.umass.edu> 2002.

[2] Apache Hadoop, The Apache foundation. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://hadoop.apache.org/>

[3] Apache Spark, The Apache foundation. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://spark.apache.org/>

Capítulo 2

[4] Eric Schmidt (Google CEO 2001-2011), Techonomy conference, Agosto 2010.

[5] Lyman, Peter y Varian, Hal R. How Much Information 2000 [en línea]. Berkeley: University of California Barkeley, 2000 [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www2.sims.berkeley.edu/research/projects/how-much-info/how-much-info.pdf>.

[6] Lyman, Peter y Varian, Hal R. How Much Information 2003 [en línea]. Berkeley: University of California Barkeley, 2003 [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://groups.ischool.berkeley.edu/archive/how-much-info-2003/printable_report.pdf>.

[7] Short, James E., Bohnm Roger E. Y Bary, Chaitanya. How Much Information 2010 [en línea]. San Diego: University of California San Diego, 2011. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://clds.sdsc.edu/sites/clds.sdsc.edu/files/pubs/ESI-Report-Jan2011.pdf>.

[8] Cisco Systems Inc. The Zettabyte Era: Trends and Analysis 2015 [en línea]. San José: Cisco Systems Inc., 2015 [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf>.

[9] The Digital Universe in 2020: Big Data, Bigger Digital Shadow s, and Biggest Growth in the Far East [en línea]. IDC Sponsored by EMC Corp. Diciembre 2012. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf>

[10] Jeffrey Dean, Sanjay Ghemaw. MapReduce: Simplified data processing on large clusters. Google, Mountain View, CA. Communications of the ACM - 50th anniversary issue: 1958 - 2008 , Volume 51, Enero 2008 Páginas 107-113.

[11] Facebook Inc., Palo Alto, CA [en línea]. [fecha de consulta: 5 Mayo 2016].
Disponible en: <https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/>

[12] Petter, Bae Brandtzæg. Big Data, for better or worse: 90% of world's data generated over last two years. ScienceDaily. Mayo, 2013.
[fecha de consulta: 5 Mayo 2016].
Disponible en: <www.sciencedaily.com/releases/2013/05/130522085217.htm>

[13] Cox, Michael y Ellsworth, David. Application-Controlled Demand Paging for Out-of-Core Visualization [en línea]. Moffett Field: NASA Ames Research Center, 1997. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www.nas.nasa.gov/assets/pdf/techreports/1997/nas-97-010.pdf>.

[14] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer. Indexing by Latent Semantic Analysis [en línea]. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf>

[15] Christos Papadimitriou, Prabhakar Raghavan and Hisao Tamaki. Latent Semantic Indexing: A Probabilistic Analysis [en línea]. 1998. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://web.stanford.edu/class/ee378b/papers/lsi.pdf>

[16] Thomas Hofmann. Probabilistic Latent Semantic Analysis. [fecha de consulta: 5 Mayo 2016].
Disponible en: <https://arxiv.org/pdf/1301.6705.pdf>

[17] David M. Blei, Andrew Y. Ng, Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (Marzo 2003), Páginas 993-1022.

[18] Rosen-Zvi, m., grif ths, t., steyvers, m., smith, P. The author-topic model for authors and documents. Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (2004), Páginas 487–494.

[19] Andrew McCallum, Xuerui Wang, Andrés Corrada-Emmanuel. Topic and Role Discovery in Social Networks. 2005.

[20] Andrew McCallum, Xuerui Wang, Andrés Corrada-Emmanuel. Topic and Role Discovery in Social Networks with Experiments on Enron and Academic Email. 2007.

[21] Laura Dietz, Steffen Bickel yTobias Scheffer. Unsupervised Prediction of Citation Influences. 2007.

[22] Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing y William W. Cohen. Joint latent topic models for text and citations. 2008. Stanford University.

[23] L. Fei-Fei y P. Perona. A Bayesian hierarchical model for learning natural scene categories. IEEE Computer Vision and Pattern Recognition. Páginas 524–531. 2005.

[24] D. Blei and M. Jordan. Modeling annotated data. In Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Página 127–134. ACM Press, 2003.

[25] E. Bart, M. Welling, and P. Perona. Unsupervised organization of image collections: Unsupervised organization of image collections: Taxonomies and beyond. Transactions on Pattern Recognition and Machine Intelligence, 2010.

[26] J. Li, C. Wang, Y. Lim, D. Blei, and L. Fei-Fei. Building and using a semantic-visual image hierarchy. In Computer Vision and Pattern Recognition, 2010

[27] H. Wallach. Topic modeling: Beyond bag of words. In Proceedings of the 23rd International Conference on Machine Learning, 2006.

[28] D. Blei and J. Lafferty. Dynamic topic models. In International Conference on Machine Learning, pages 113–120, New York, NY, USA, 2006. AC

[29] D. Blei, T. Griffiths, and M. Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. Journal of the ACM, 57. Páginas: 1–30, 2010.

[30] David Blei. Latent Dirichlet Allocation. C variational inference implementation. Princeton University [en línea]. 2003. [fecha de consulta: 5 Mayo 2016].
Disponible en: <https://www.cs.princeton.edu/~blei/lda-c/index.html>

[31] Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung. The Google File System. Google, Mountain View, CA. 2003.

[32] Yahoo! Launches World's Largest Hadoop Production Application (Hadoop and Distributed Computing at Yahoo!) [en línea].
[fecha de consulta: 5 Mayo 2016].
Disponible en:
<http://web.archive.org/web/20080514135459/http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html>

[33] Powered by Apache Hadoop [en línea]. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://wiki.apache.org/hadoop/PoweredBy>

[34] Paris Carbone, Gyula Fóra, Stephan Ewen, Seif Haridi, Kostas Tzoumas. Lightweight Asynchronous Snapshots for Distributed Dataflows. KTH Royal Institute of Technology. Junio 2015.

[35] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker e Ion Stoica. Spark: Cluster Computing with Working Sets. University of California, Berkeley.

Capítulo 3

[36] "2246 documents from the Associated Press" [en línea].
[fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www.cs.princeton.edu/~blei/lda-c/ap.tgz>

[37] HewlettPackard HP ProLiant DL380 G5 High Performance Datasheet [en línea].
[fecha de consulta: 5 Mayo 2016].
Disponible en: <http://www.ossistemes.com/pdf/dl380.pdf>

Capítulo 4

[38] Hadoop 2.7.1 Release Notes. Apache Foundation [en línea].
[fecha de consulta: 5 Mayo 2016].
Disponible en: <http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/releasenotes.html>

[39] Hadoop-Lda [en línea]. [fecha de consulta: 5 Mayo 2016].
Disponible en: <http://code.google.com/archive/p/Hadoop-lda>

[40] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, Mohamad Alkhouja. Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce. University of Maryland. University of Maryland. 2012.

[41] Anaconda Suite [en línea]. Continuum Analytics.
[fecha de consulta: 5 Mayo 2016].
Disponible en: <https://www.continuum.io/downloads>.

[42] D3.js, Data-Driven Documents [en línea]. Mike Bostock, Jeffrey Heer, Vadim Ogievetsky. [fecha de consulta: 5 Mayo 2016].
Disponible en: <https://d3js.org/>.

[43] F. Yan, N. Xu, and Y. Qi, "Parallel inference for latent dirichlet allocation on graphics processing units," GPU-LDA. Proceedings of Advances in Neural Information Processing Systems, 2009, páginas 2134–2142.

[44] A. Asuncion, P. Smyth, and M. Welling, "Asynchronous distributed learning of topic models," Async-LDA. Proceedings of Advances in Neural Information Processing Systems, 2008.

[45] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang, "PLDA: parallel latent Dirichlet allocation for large-scale applications," in International Conference on Algorithmic Aspects in Information and Management, 2009.

[46] A. J. Smola, S. Narayanamurthy. "An architecture for parallel topic models". Y!LDA. International Conference on Very Large Databases, vol. 3, 2010.

[47] A. S. Foundation, I. Drost, T. Dunning, J. Eastman, O. Gospodnetic, G. Ingersoll, J. Mannix, S. Owen, K. Wettin. Apache Mahout. 2010. [fecha de consulta: 5 Mayo 2016]. Disponible en: <http://mloss.org/software/view/144/>

[48] Large Scale Topic Modeling: Improvements to LDA on Apache Spark. . [fecha de consulta: 5 Mayo 2016].
Disponible en: <https://databricks.com/blog/2015/09/22/large-scale-topic-modeling-improvements-to-lda-on-apache-spark.html>

[49] Matthew D. Hoffman, David M. Blei, Francis Bach. Online Learning for Latent Dirichlet Allocation. 2010.

[50] Arthur Asuncion, Max Welling, Padhraic Smyth, Yee Whye Teh. 2009. On smoothing and inference for topic models. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09). AUAI Press, Arlington, Virginia, United States, 27-34.

Capítulo 5

[51] "El número de suicidios duplica al de los muertos por accidentes de tráfico", El País. 31 Marzo 2016. [fecha de consulta: 5 Mayo 2016].
Disponible en:
<http://politica.elpais.com/politica/2016/03/29/actualidad/1459249694_040134.html>

Apéndice A

Tabla de Stopwords

En este apéndice se muestran todas las palabras sin significado léxico que se han eliminado del corpus en el proceso de pre-filtrado.

a
abajo
acá
acaso
además
adonde
adónde
adrede
ahí
ahora
al
algo
alguien
alguna
algunas
alguno
algunos
allá
allí
alrededor
ante
antes
apenas
aprisa
aquel
aquella
aquellas
aquello
aquellos
aquí
arriba
así
asimismo
atrás
aunque
aún
ayer
bajo
bastante
bien
cabe
cada
casi
cerca
cierto
claro
como
cómo
con
conmigo
conque
consigo
contigo
contra
creo
cree
creen
cual
cuál
cuáles
cualesquiera
cualquiera
cuando
cuándo
cuánto
cuanto
cuya
cuyas
cuyo
cuyos
de
debajo
del
delante
demás
demasiada
demasiadas
demasiado
demasiados
deprisa
desde
despacio
después
detrás
dice
donde
dónde
durante
e
el
él
ella
ellas
ello
ellos
en
encima
enfrente
enseguida

entre
entonces
era
erais
éramos
eran
eras
eres
es
esa
esas
escasa
escasas
escaso
escasos
ese
eso
esos
esta
está
estaba
estabais
estábamos
estaban
estabas
estad
estada
estado
estados
estáis
estamos
están
estando
estar
estará
estarán
estarás
estaré
estaréis
estaremos
estaría
estaríais
estaríamos
estarían
estarías
estas
estás
este
esté
estéis
estemos
estén
estés
esto
estos
estoy
estuve
estuviera
estuvierais
estuviéramos
estuvieran
estuvieras
estuvieron
estuviese
estuvieseis
estuviésemos
estuviesen
estuvieses
estuvimos
estuviste
estuvisteis
estuvo
exacto
excepto
fue
fuera
fuerais
fuéramos
fueran
fueras
fueron
fuese
fueseis
fuésemos
fuesen
fueses
fui
fuimos
fuiste
fuisteis
ha
habéis
haber
había
habíais
habíamos
habían
habías
habido
habidos
habiendo
habrá
habrán
habrás
habré
habréis
habremos
habría
habríais
habríamos
habrían
habrías
hace
hacen
hacia
hacer
hacían
hacía
haremos
hiciese
hiciesen
hagan
haga
han
has
hasta
hay
haya
hayáis
hayamos
hayan
hayas
he
hemos
hecho
hoy
hube
hubiera

hubierais
hubiéramos
hubieran
hubieras
hubieron
hubiese
hubieseis
hubiésemos
hubiesen
hubieses
hubimos
hubiste
hubisteis
hubo
inclusive
incluso
jamás
la
las
le
lejos
les
lo
los
luego
mal
mañana
más
me
mediante
mejor
menos
mi
mí
mía
mías
mientras
mío
míos
mis
misma
mismas
mismo
mismos
mucha
muchas
mucho
muchos
muy
nada
nadie
ni
ninguna
ningunas
ninguno
ningunos
no
nos
nosotras
nosotros
no_obstante
nuestra
nuestras
nuestro
nuestros
nunca
o
obvio
os
otra
otras
otro
otros
para
peor
pero
pese
poca
pocas
poco
pocos
por
porque
pronto
puede
pueden
puedo
pues
puesto
que
qué
quien
quién
quienes
quiénes
quienesquiera
quienquiera
quiero
quiere
quizá
quizás
regular
salvo
saben
se
sea
seáis
seamos
sean
seas
según
ser
será
serán
serás
seré
seréis
seremos
sería
seríais
seríamos
serían
serías
si
sí
sido
siempre
siendo
sin
sin_embargo
sino
siquiera
so
sobre
sois
solamente
solo
somos

son
soy
su
sus
suya
suyas
suyo
suyos
tal
también
tampoco
tan
tanta
tantas
tanto
tantos
tarde
te
temprano
tendrá
tendrán
tendrás
tendré
tendréis
tendremos
tendría
tendríais
tendríamos
tendrían
tendrías
tened
tenéis
tenemos
tener
tenga
tengáis
tengamos
tengan
tengas
tengo
tenía
teníais
teníamos
tenían
tenías
tenido
teniendo
the
ti
tiene
tienen
tienes
toda
todas
todavía
todo
todos
tras
tu
tú
tus
tuve
tuviera
tuvierais
tuviéramos
tuvieran
tuvieras
tuvieron
tuviese
tuvieseis
tuviésemos
tuviesen
tuvieses
tuvimos
tuviste
tuvisteis
tuvo
tuya
tuyas
tuyo
tuyos
u
último
última
últimas
últimos
un
una
unas
uno
unos
usted
ustedes
varias
varios
viceversa
viene
vienen
vos
vosotras
vosotros
voy
vuestra
vuestras
vuestro
vuestros
y
ya
yo

Apéndice B

Modelos LDA

En este apéndice se mostrarán los modelos LDA de cada tema, generados a partir de los experimentos expuestos en el Capítulo 5, empleando los valores de $VocabSize$ y $numTopics$ que han maximizado la tasa de acierto de cada uno de ellos.

B.1 Modelo LDA para el tema de Cultura.

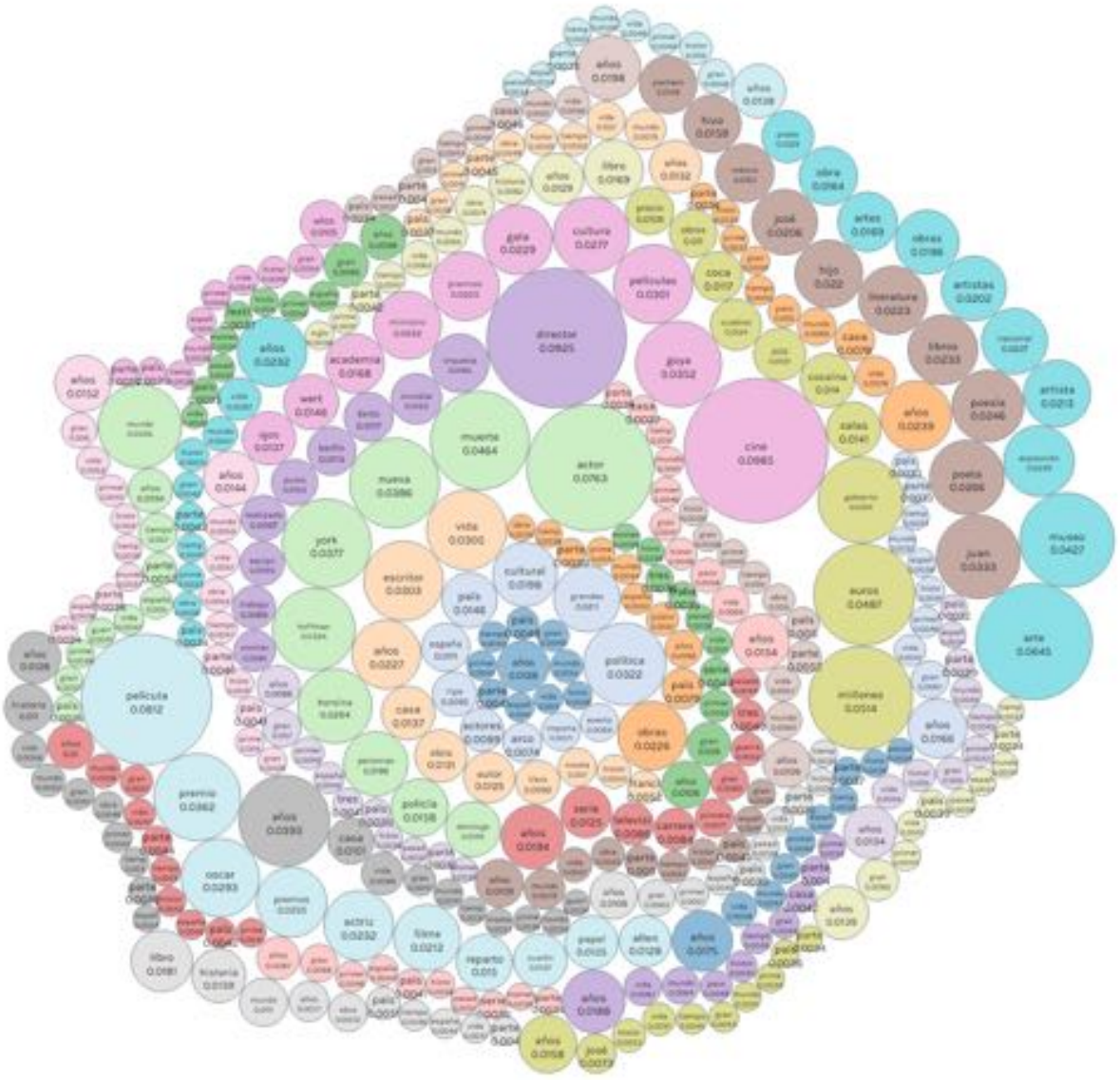


Figura B.1: Gráfica clusterizada de los subtemas latentes hallados en la sección de Cultura del Corpus para $VocabSize = 3000$ y $numTopics = 40$.

B.2 Modelo LDA para el tema de Economía.

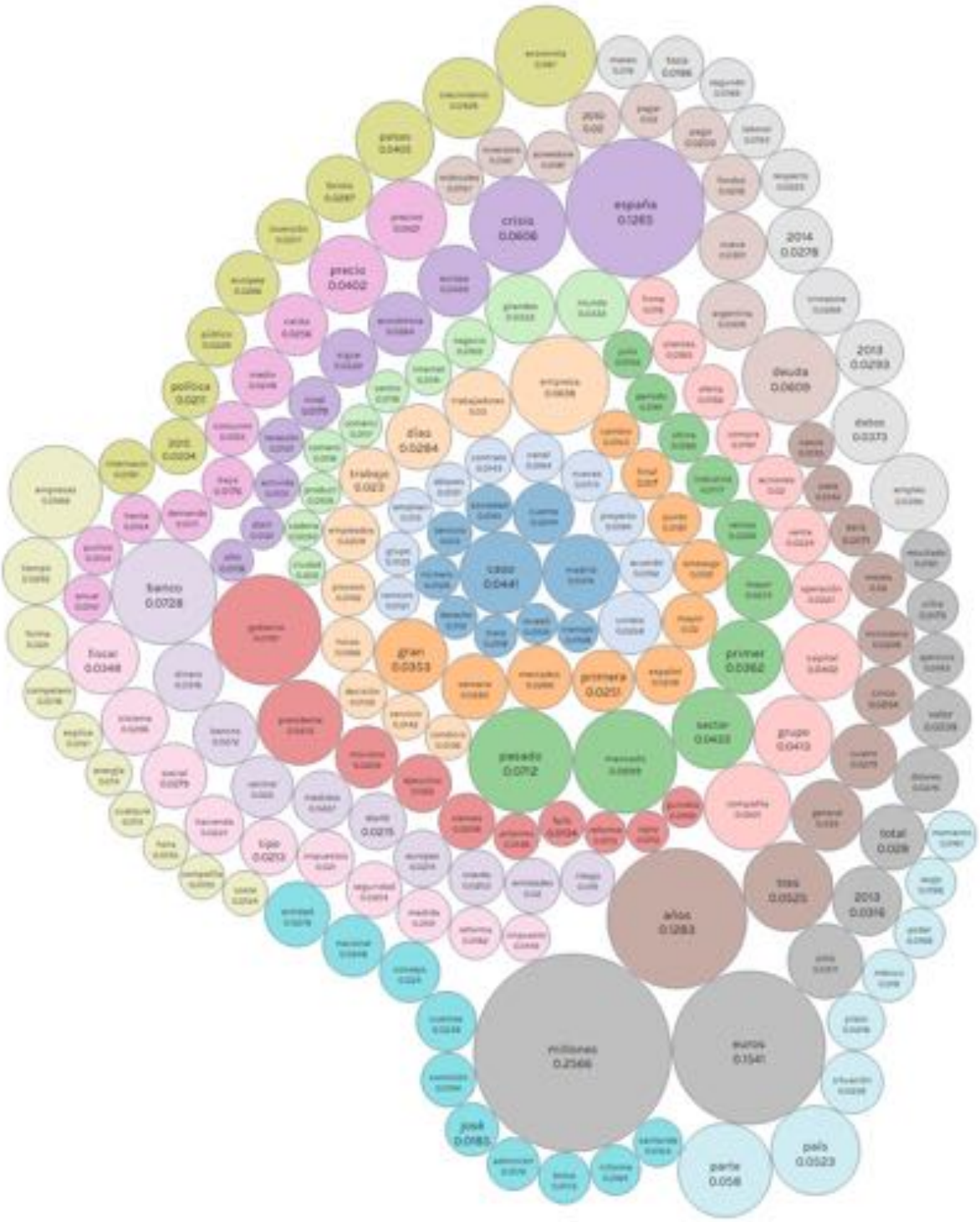


Figura B.2: Gráfica clusterizada de los subtemas latentes hallados en la sección de Economía del Corpus para $VocabSize = 5000$ y $numTopics = 20$.

B.3 Modelo LDA para el tema Internacional.

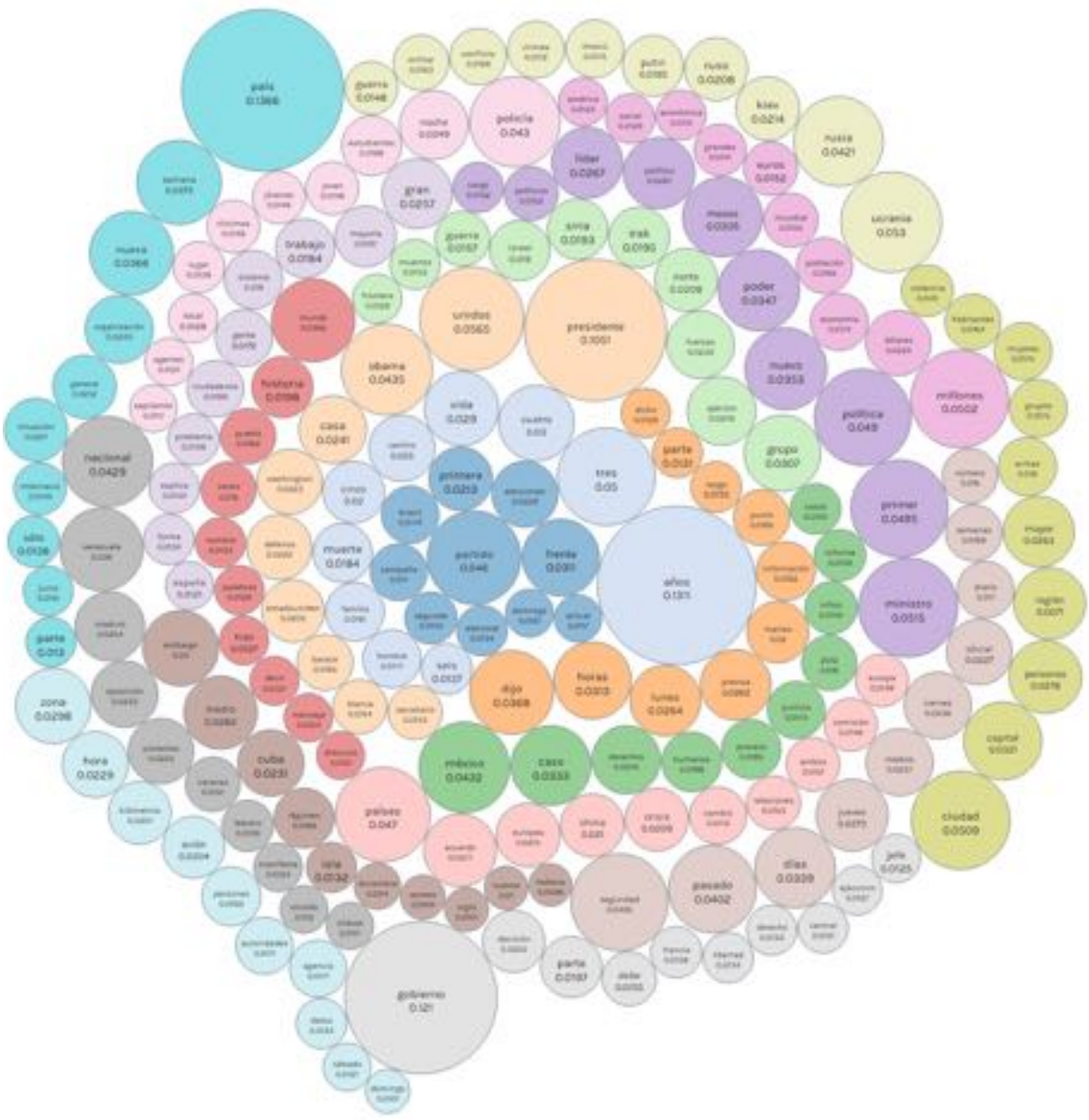


Figura B.3: Gráfica clusterizada de los subtemas latentes hallados en la sección Internacional del Corpus para $VocabSize = 5000$ y $numTopics = 20$.

B.4 Modelo LDA para el tema Nacional.

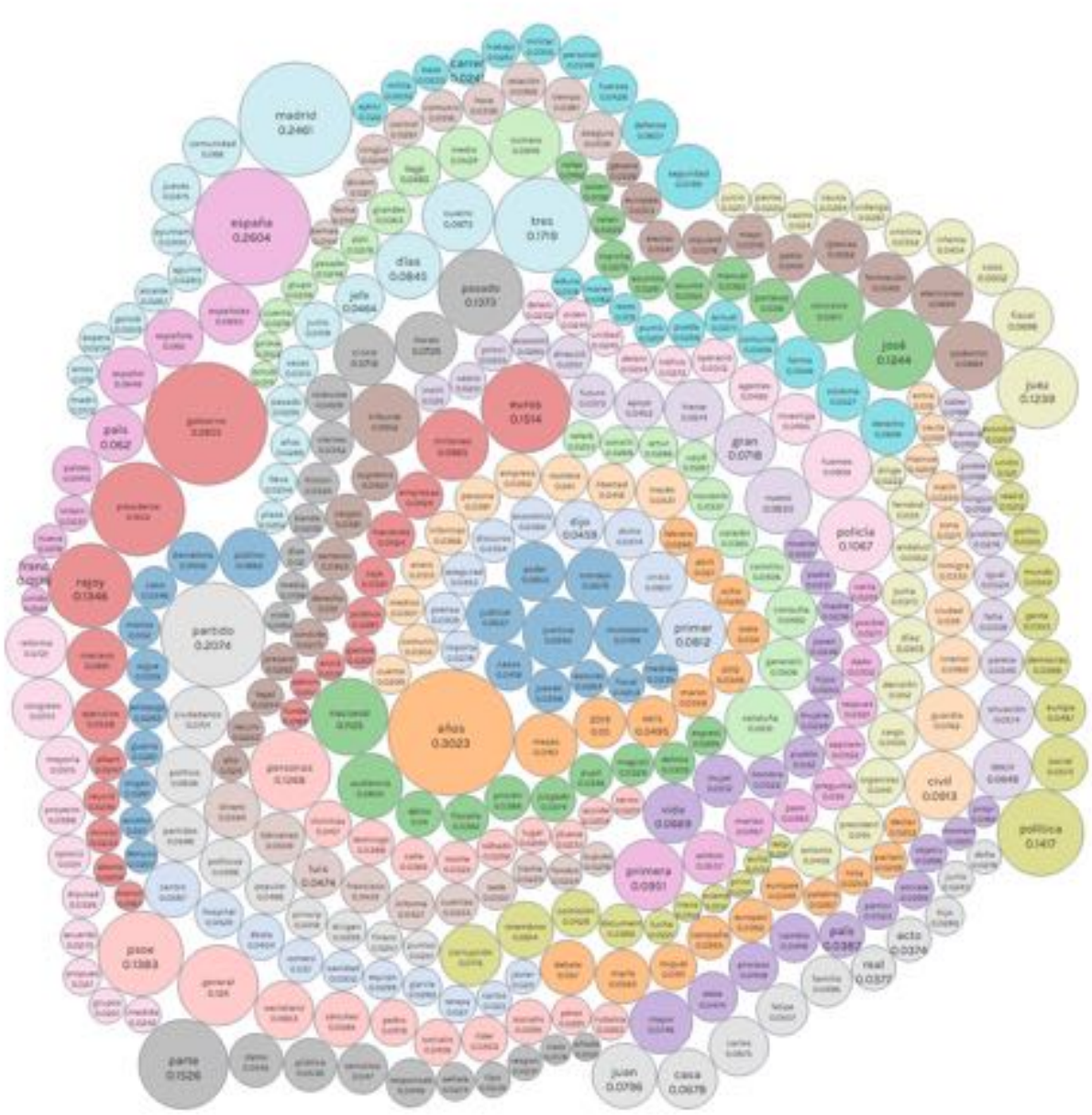


Figura B.4: Gráfica clusterizada de los subtemas latentes hallados en la sección Nacional del Corpus para $VocabSize = 2000$ y $numTopics = 40$.

B.5 Modelo LDA para el tema de Sociedad.

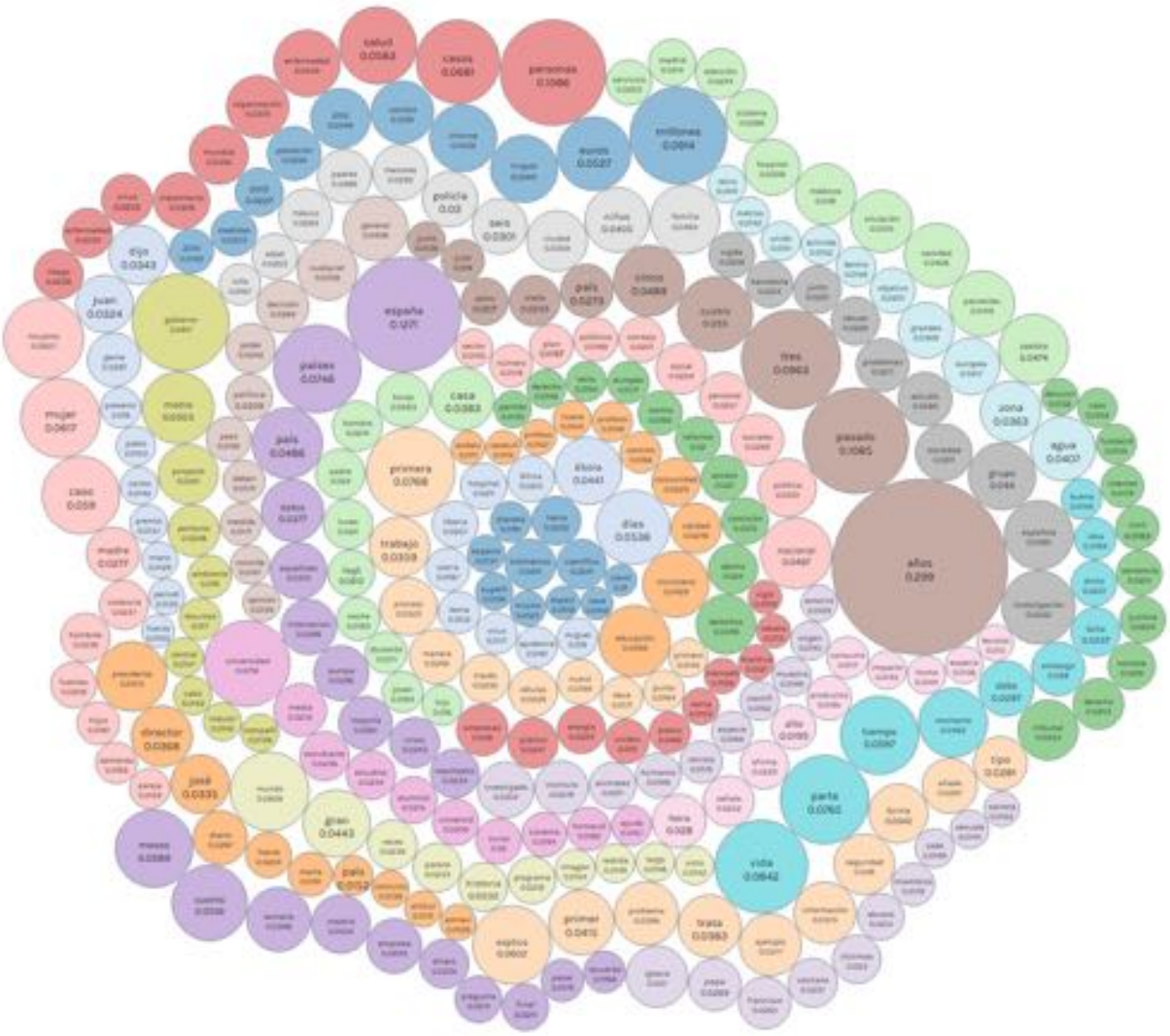


Figura B.5: Gráfica clusterizada de los subtemas latentes hallados en la sección de Sociedad del Corpus para $VocabSize = 4000$ y $numTopics = 30$.

B.6 Modelo LDA para el tema de Deportes.

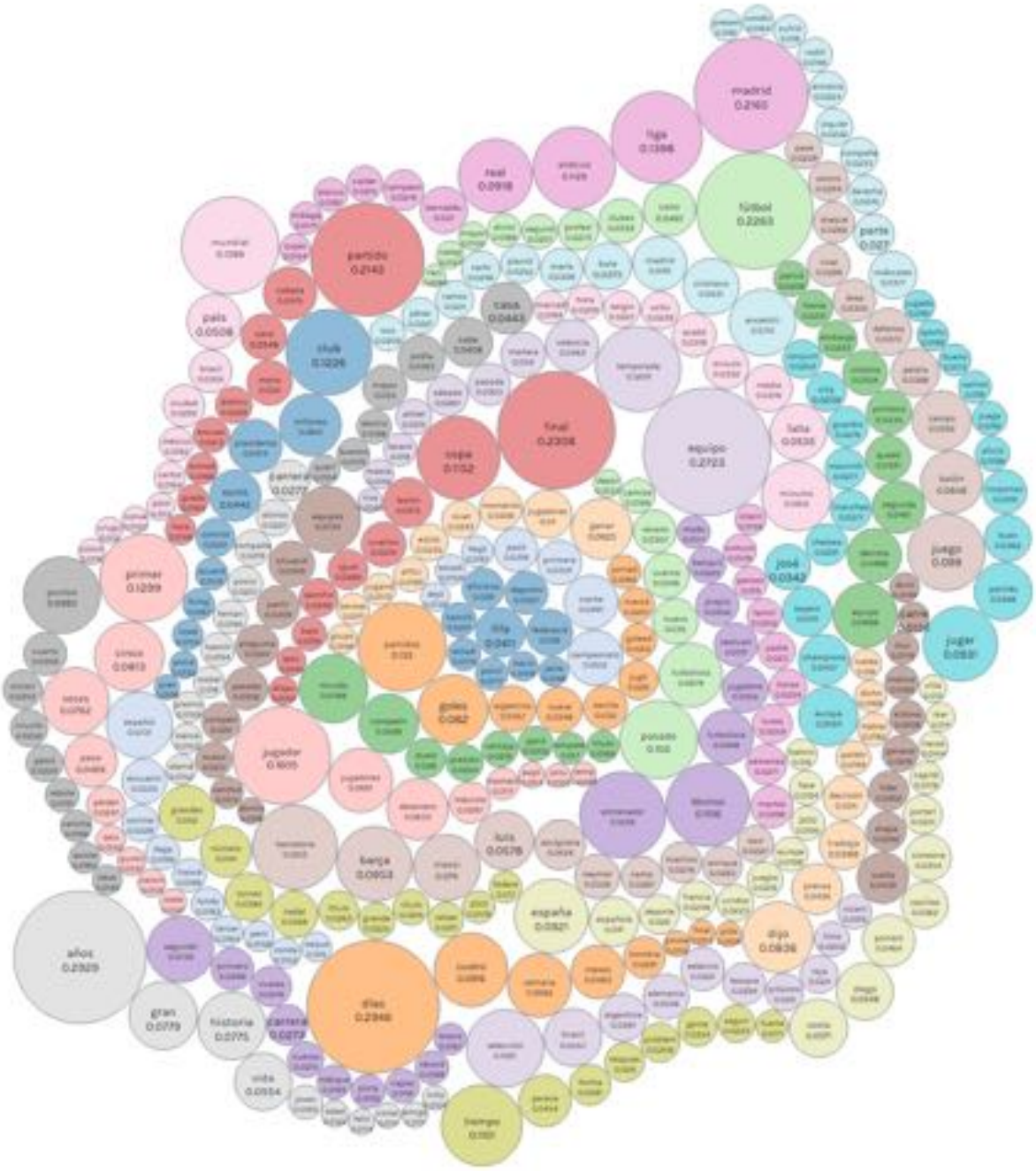


Figura B.6: Gráfica clusterizada de los subtemas latentes hallados en la sección de Deportes del Corpus para $VocabSize = 4000$ y $numTopics = 40$.

B.7 Modelo LDA para el tema de Tecnología.

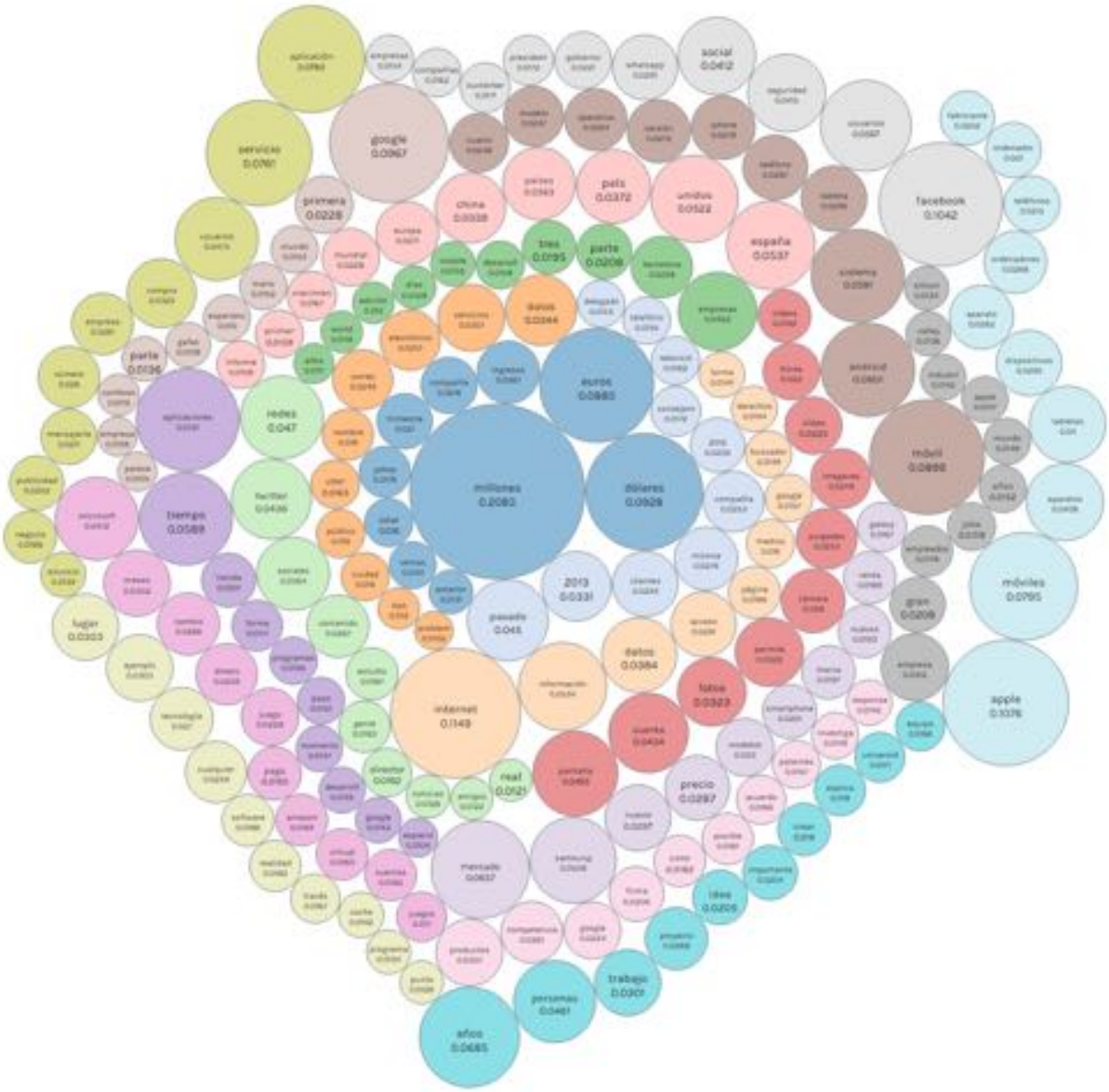


Figura B.7: Gráfica clusterizada de los subtemas latentes hallados en la sección de Tecnología del Corpus para $VocabSize = 2000$ y $numTopics = 20$.

Apéndice C

Presupuesto

Ejecución Material

- Compra de servidor HP ProLiant DL160

  de 6 núcleos a 2.4 GHz, 16 Gigabytes de RAM…….…….. 1.625,00 €
- Material de oficina ……..…… 200,00€
- Total de Ejecución Material……..…….…… 1.825,00€

Gastos Generales

- 16 % sobre Ejecución Material……..……..…… 292,00€

Beneficio Industrial

- 6 % sobre Ejecución Material……..……..……..…..… 109,50€

Honorarios Proyecto

- 960 horas a 15€/hora ……..……..……..…..… 14.400,00€

Material fungible

- Gastos de impresión ……..……..……..…..…100,00€
- Gastos de encuadernación ……..……..……..……..…..…30,00€

Subtotal del Presupuesto

- Subtotal del Presupuesto ……..……..…16.756,50€

IVA Aplicable

- 21% Subtotal Presupuesto ……..……..…. 3.518,86€

Total Presupuesto

- Total Presupuesto ……..……..…….. 20.275,36 €

Madrid, Mes de Junio de 2016
El Ingeniero Jefe de Proyecto

Fdo.: Bruno Stampete
Ingeniero de Telecomunicación

Apéndice D

Pliego de Condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un estudio de **TOPIC MODELING y BIG DATA** para noticias en prensa. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del

Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por

agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.