UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Manejo de herramientas Big Data para realizar topic modeling en discursos universitarios y clusterización de los resultados

Alejandro Aguirre Tamaral
Tutor: Estrella Pulido Cañabate

Enero 2016

Manejo de herramientas Big Data para realizar topic modeling en discursos universitarios y clusterización de los resultados

AUTOR: Alejandro Aguirre Tamaral
TUTOR: Estrella Pulido Cañabate

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

Resumen

Los motivos de un estudiante para elegir ingresar en una universidad entre las múltiples existentes pueden resultar difíciles de cuantificar. La principal razón de dicha elección, aparte de la distancia a la que se encuentra, es comprensible que sea debida a la reputación de la universidad elegida. Existen diversos rankings que clasifican a las universidades más prestigiosas, basándose en ciertas características que éstas poseen. Por lo tanto, existen atributos que hacen que una universidad pueda obtener una mejor clasificación que otra (según el criterio seguido por el ranking en cuestión).

La hipótesis de este TFG es estudiar si otro atributo para clasificar universidades podría basarse en los temas que abordan los rectores de dichas universidades en los discursos que imparten.

En este Trabajo de Fin de Grado se ha realizado tanto el proceso de recolección de dichos discursos como el manejo de éstos para la obtención de resultados. De esta forma, se pueden ajustar los parámetros de la búsqueda según el requerimiento del análisis. Además, al conocer los datos, éstos se pueden pre-procesar para optimizar los resultados obtenidos por los algoritmos de análisis de texto. Centrándose en la temática de los discursos, en primer lugar hay que seguir una metodología para obtener múltiples discursos de las universidades que se quieran analizar. Se mezclan diversos métodos de recolección de datos para obtener el mayor número posible de muestras que analizar. Una vez obtenidos y validados los discursos, se pueden analizar con diversos algoritmos de análisis de texto. Tras ejecutar un algoritmo que halla los temas de cada discurso, se pueden buscar diferencias entre los discursos de universidades con un ranking destacado y los de universidades con un ranking inferior. De esta forma, es posible comprobar si la diferencia de clasificación entre diversas universidades es apreciable mediante un recurso tan accesible como lo es el discurso de un rector. Además de por los temas, se puede intentar diferenciar la clasificación de una universidad por la cantidad de discursos que aloja en su página web y por la facilidad en la que éstos se pueden descargar (debido a que están bien ordenados en la página web).

Palabras clave

Araña Web, Análisis de texto, Clusterización, Discursos, Python

Abstract

The reasons for a student to enrol in a certain university amongst all the possibilities, could be difficult to quantify. The main argument for this choice, apart from the distance to the university, is reasonable to be the chosen university reputation. There are different rankings that classify the most prestigious universities, based on different features. Therefore, there are some characteristics that make a university to have a better classification than another (based on the ranking criteria).

This Bachelor Thesis is based on the hypothesis that the topics included in the speeches delivered by University rectors can be used as an additional indicator to classify universities.

The work includes both the process of crawling for obtaining the speeches and the handling of this data to acquire results. Therefore, the seeking process could be adjusted by the analysis requirements. Moreover, since data are known, it is possible to pre-process them for optimizing the results obtained by the text analytics algorithms. By focusing on the topics issue, the first step is to follow a methodology for collecting multiple speeches of the universities to be analysed. Diverse methods are mixed for obtaining the maximum number of samples to analyse. Once the speeches have been acquired and validated, they can be analysed by applying various text analytics algorithms. After running an algorithm that discovers the topics of each speech, it is possible to look for differences between the speeches of the universities with a high classification and the ones with a lower classification. Thus, it is possible to research whether the classification of a university is also appreciable by a free resource as its rector speeches. In addition to the topics, another way to try to classify a university is by the amount of speeches that they upload in their web pages or by the easiness to obtain these speeches (due to how well organized is the web page).

Keywords

Web Crawler, Text Analytics, Topic Modeling, Clustering, Speeches, Python

Agradecimientos

En primer lugar quiero agradecer a mi familia las oportunidades y el apoyo ofrecido durante la carrera. Además, me gustaría agradecer la oportunidad de realizar este TFG a la Cátedra UAM/IBM y sobre todo, a mi tutora Estrella, por la constancia en la ayuda y la supervisión durante todo el proceso de realización del trabajo.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

La idea de este proyecto surge a partir de una colaboración de la Cátedra UAM/IBM con la Facultad de Económicas de la UAM. La motivación personal ha sido la de realizar un proyecto en el cual poder aportar los conocimientos adquiridos durante el grado, además de poder hacerlo en una cátedra universitaria.

1.1 Motivación

Existen diversas universidades que poseen un gran reconocimiento a nivel mundial y que, usualmente, se posicionan entre las primeras en cualquier ranking internacional de universidades. Se utilizan distintos atributos para evaluar si una universidad tiene una mejor reputación o un mejor puesto en un ranking que otra. Dichos atributos podrían, por ejemplo, ser la metodología que siguen para la enseñanza o la formación que reciben sus estudiantes.

Los temas sobre los que tratan los discursos del rector de una determinada universidad indican las prioridades de esa universidad en cuanto a investigación, docencia, etc. Dichos temas también podrían utilizarse para evaluar el prestigio de una determinada universidad comprobando si existe cierta semejanza entre los temas que aparecen en los discursos de universidades con una posición similar en el ranking.

Por lo tanto, la motivación de este proyecto ha sido seguir una metodología que permita obtener múltiples discursos (de la manera más automática posible), analizar los temas sobre los que tratan y comprobar si existe una correlación entre dichos temas y el lugar que ocupan las respectivas universidades en los rankings internacionales.

1.2 Objetivos

Los objetivos de este trabajo han sido:

- Creación de un repositorio textual que contuviera discursos de rectores de diversas universidades presentes en un ranking, para posteriormente ser analizados.

- Empleo de algoritmos de análisis de texto (Topic modeling) para obtener los temas de los que tratan dichos discursos.

- Utilización de diversos algoritmos que busquen relaciones a partir de los temas de los cuales tratan los discursos, y de esta forma comprobar si existen ciertos patrones entre discursos con un ranking similar.

- Análisis de diversos programas para cada fase, escogiendo el que más se ajuste a las características de este trabajo. Además, se ejecutarán programas que sigan el paradigma de computación en la nube y se compararán sus resultados con los programas con la misma funcionalidad que se ejecuten de forma local.

- La metodología que se emplee ha de poseer un procedimiento general, es decir, que se pueda aplicar toda la metodología a otros archivos que no sean discursos de rectores.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estudio del estado del arte:** en este capítulo se expondrá el contexto en el que se sitúan tanto los métodos que se van a emplear, como las herramientas que se manejarán para ello.

- **Desarrollo:** en este capítulo se explicará la metodología con la que se han aplicado las diversas herramientas analizadas. Se explicará el proceso empleado, comenzando por la etapa de web crawling. Posteriormente se indicarán las herramientas empleadas para realizar text analytics, detallando la herramienta manejada para ejecutar algoritmos de topic modeling. En general, se indicarán qué programas (y cómo) se han empleado y se argumentarán los motivos por los que se ha descartado el uso de diversos programas con funcionalidades similares a los empleados.

- **Pruebas y resultados:** en este capítulo se explicarán las pruebas realizadas, aplicando la metodología explicada en el apartado de desarrollo a la temática de los discursos de rectores. Además se explicarán los resultados obtenidos en cada etapa.

- **Conclusiones y trabajo futuro:** en este capítulo se resumirán las ideas obtenidas tras la conclusión del trabajo y cómo han sido resueltos los objetivos establecidos al comienzo del mismo. Además se indicarán las posibles líneas de trabajo futuro.

2 Estado del arte

En este capítulo se describen los aspectos más relevantes acerca de los temas sobre los que trata el trabajo.

2.1 Web crawling

Existe una modalidad de programa, denominada araña web (web crawler), cuya función es el rastreo automático de páginas web. Estos programas, a partir de unos enlaces base que se indican previamente a la ejecución, obtienen sucesivamente nuevos enlaces con los que continuar el rastreo mientras se cumplan las condiciones establecidas por el usuario. Estos programas se configuran mediante diversos parámetros de entrada y devuelven como salida, los archivos encontrados. El formato en el que se devuelven los resultados depende del programa. Los parámetros de entrada más usuales son:

- Los enlaces con los que comenzar el análisis.

- Los parámetros que causan la finalización del programa, tales como la profundidad máxima permitida (el nivel de profundidad de un enlace es el número de enlaces intermedios que han de descargarse para llegar al mismo), el número máximo de enlaces analizados o el tiempo máximo de ejecución.

- La cadena de texto que ha de contener un enlace para que sea descargado o no (filtro). Los filtros se colocan junto a un símbolo, que puede ser el símbolo '+' (indica que esa cadena ha de aparecer en el enlace para que sea descargado) o el símbolo '–' (indica que la cadena debe no aparecer). Si no se especifica símbolo, el '+' es, usualmente, el símbolo predeterminado. Aunque existan múltiples filtros en el crawler, únicamente es necesario que el enlace cumpla un filtro '+' para que se descargue, o uno '-' para que se rechace.

2.1.1 Componentes

Aunque existan diferentes tipos de crawlers, y cada programa pueda ofrecer utilidades propias, se pueden destacar unos componentes básicos comunes [1] que se enumeran a continuación.

- **Frontera del crawler**. Su función es albergar la lista de enlaces pendientes de ser rastreados, y seleccionar, cada vez, el próximo enlace que se descargará. Una vez se ha descargado el contenido de una página web, es probable que se hayan encontrado nuevos enlaces (que cumplan los filtros), los cuales serán, a su vez,

añadidos a la frontera y quedarán a la espera de ser seleccionados para ser descargados.

- **Descargador de páginas**. Este componente se encarga de descargar las páginas web que le indica la frontera. Funciona como un cliente HTTP que envía peticiones e interpreta las respuestas del servidor. Dependiendo de los parámetros indicados al crawler, este componente puede cancelar la descarga de un enlace si se excede en tiempo o en tamaño.

- **Repositorio**. Es el espacio en el que se guardan los archivos descargados.

2.1.2 Modos de ejecución

Los crawlers se pueden clasificar en cuatro categorías de acuerdo con el modo en el que se ejecutan.

- **Crawlers centrados**. Su objetivo principal es descargar páginas relacionadas con un tema concreto. Por tanto, es necesario hallar la relevancia respecto del tema de cada página que se visita para determinar si será descargada. El principal problema de este método radica en conocer qué páginas serán similares al objetivo sin haberlas descargado. La ventaja es que minimiza considerablemente el número de páginas que se visitan, por lo que es el más adecuado si no se dispone de un hardware que permita realizar búsquedas avanzadas y se pretenden encontrar archivos pertenecientes a un determinado tema. La primera opción para manejar este método consiste en ejecutar un crawler centrado por definición, y que, mediante un algoritmo, escoja de la frontera, con prioridad, las páginas con más relevancia respecto del tema en el que se centra el crawler. Otra opción consiste en ejecutar un crawler tradicional añadiéndole filtros. Por ejemplo, al emplear el filtro ".jpg", el crawler se centraría en descargar las imágenes con formato JPG que estén contenidas en los enlaces base.

- **Crawlers incrementales**. Un crawler se puede ejecutar en modo incremental, una vez que el crawler completo ha finalizado, para mantener los archivos del repositorio actualizados. La diferencia radica en que en el modo tradicional se repetiría todo el proceso de rastreo para que los resultados estén actualizados, mientras que en éste, la actualización de resultados se realiza de forma incremental visitando las páginas frecuentemente, mediante un algoritmo que indica qué páginas hay que actualizar. Existen diversas medidas [2] para estimar el momento de realizar las actualizaciones, tales como el tiempo que ha transcurrido desde que el enlace entró en el repositorio o el tiempo que ha transcurrido desde que el contenido de dicho enlace fue modificado en la página web de origen.

- **Crawlers paralelos**. Debido al notable volumen de enlaces que alberga la frontera, estos programas son muy paralelizables. Dicha paralelización resulta necesaria cuando se realiza una búsqueda con una cantidad de enlaces desmesurada. Este modo se emplea en diversos motores de búsqueda, ya que no son crawlers centrados en un tema, sino que realizan un rastreo general para analizar la mayor parte posible de la Web y de esta forma poder devolver al usuario de forma precisa las páginas web que más se asemejen con la consulta que se haya introducido.

- **Crawlers distribuidos**. Son los crawlers que siguen el modelo de computación distribuida durante su ejecución, la cual permite que un programa se ejecute en diversas áreas que no están conectadas físicamente. Este método, junto con la paralelización, posibilita la ejecución de grandes búsquedas sin poseer físicamente un equipo con la suficiente capacidad para ello. Se emplea en diversos motores de búsqueda para realizar consultas de una gran dimensión de forma paralela en un tiempo viable. Desde hace años, los crawlers más importantes (GoogleBot, Yahoo! Slurp…) lo han empleado. YaCy es un ejemplo de programa de software libre que se ejecuta de forma distribuida mediante una infraestructura P2P.

2.1.3 Políticas de ejecución

Debido a que los crawlers son programas que se ejecutan de manera automática, las decisiones en tiempo de ejecución también han de resolverse automáticamente. Por ello, existen diferentes políticas que describen las diversas opciones posibles de comportamiento de un crawler. Las políticas se eligen antes de la ejecución. Dichas políticas son las siguientes:

- **Elección de enlace en la frontera del crawler para ser rastreado**. Pueden otorgarse prioridades a los enlaces, escogerse por orden de llegada, o simplemente de manera aleatoria. Debido al enorme número de enlaces que puede albergar la frontera, y a que éstos pueden crecer de manera exponencial hasta que el programa cumpla alguna de las condiciones de parada, resulta adecuado establecer un protocolo en el que se asignen prioridades a determinados enlaces. Un ejemplo de algoritmo que establece un ranking entre los enlaces es PageRank, el cual ha sido uno de los algoritmos empleado por el crawler de Google (GoogleBot). Este algoritmo asigna a cada enlace un número entre 0 y 10 que indica el ranking que éste posee, en función de la cantidad de páginas que enlazan con dicho enlace y, a su vez, del ranking de esas páginas. La función que se utilizó en el algoritmo inicial aparece en la Figura 1.

$$\mathrm{PR}(A) = (1-d) + d\sum_{i=1}^{n} \frac{\mathrm{PR}(i)}{C(i)}$$

Figura 1. Función básica del algoritmo PageRank

Los parámetros de la Figura 1 tienen el siguiente significado: PR(A) es el ranking del enlace A, d es un factor de regulación que indica la probabilidad de que el usuario continúe navegando entre los enlaces de la página en la que se encuentra, PR(i) es el ranking de cada página i que llama al enlace A, y C(i) es el número total de enlaces salientes de cada página i.

PageRank (algoritmo privado) se basó en el modelo del SCI (Science Citation Index), el cual fue elaborado en la década de los 50 para asignar méritos científicos a partir del número de publicaciones de cada científico y el número de referencias que éste obtenía en otras publicaciones. Una variante del PageRank es el HITS, que califica una página en base a la calidad de la información que posee junto con la calidad de las páginas con las que ésta enlaza. Otro posible método podría ser escoger enlaces en función del ranking que posea la página web (por ejemplo en el ranking de Alexa).

- **Decisión de si una página web es descargada o no**. Se pueden emplear filtros para indicar al crawler si descargar o rechazar cada página web. Además de los filtros, hay otros factores que pueden causar el rechazo de un enlace por parte del crawler, tales como el tiempo máximo de respuesta o el tamaño máximo del archivo a descargar. En el modo centrado, esta política adquiere importancia para descartar las páginas que no sean relevantes para el tema principal de la búsqueda.

- **Política de paralelización**. Es necesaria en los crawlers que se ejecuten en paralelo para mantener el orden del programa y que no ocurran situaciones como que se rastree el mismo enlace en más de una ocasión. De este modo, tanto la frontera como la lista de enlaces analizados deben ser comunes a todos los procesos.

- **Política de volver a rastrear**. Se aplica en busca de la actualización de los resultados del repositorio. Se realiza una estimación del momento en el que cada enlace ha de ser actualizado.

- **Política de cortesía**. Diversos crawlers pueden analizar simultáneamente una página web. Si cada uno realiza demasiadas peticiones al servidor, éste puede quedar colapsado. Para evitar dicha situación, los administradores de la página web comunican, en un archivo denominado "robots.txt", las acciones de cortesía que esperan que los crawlers apliquen. Dicho archivo está alojado en el directorio de nivel superior de la página web, e indica a los crawlers las secciones de la web que pueden ser rastreadas y las que no. También puede indicar los nombres de los crawlers que se permite que rastreen y de los que no. Se puede emplear el símbolo '*' para referirse a todos los crawlers o a todas las secciones de la página. Además, es posible indicar el tiempo mínimo que ha de pasar entre dos peticiones de un mismo crawler, para evitar que un bot colapse el servidor y propicie que la web deje de estar operativa para los usuarios. El siguiente ejemplo negaría el acceso de todos los crawlers a cualquier ruta de la web que comience por '/' (la raíz de la

página), por lo que ninguna ruta de esa web puede ser accedida de forma automática por ningún crawler:

*User-agent: **
Disallow: /

Si una página web no posee este archivo, el crawler puede emplear su propia política de cortesía predeterminada, o no seguir ninguna. Es más, aunque la página web posea el archivo, un crawler puede no cumplir dichas instrucciones, ya que se trata de un archivo informativo, por lo que podría rastrear las rutas que el archivo indica que no deberían ser rastreadas. De hecho, pueden existir rutas de la web no visibles al usuario, y al citarlas en el robots.txt, produzcan el efecto de informar sobre la existencia de dichas rutas y sobre el hecho de que se pretende que permanezcan ocultas. Por tanto, el efecto del fichero sería contraproducente. En cualquier caso, este archivo puede hacer que el número de resultados decrezca.

2.1.4 Crawlers analizados

Existen diversos programas que han sido desarrollados para ejecutar técnicas de web crawling y que están catalogados como software libre. No obstante, las características de muchos de esos programas son similares [3]. A continuación se analizan cuatro crawlers que poseen distintas características:

- **Web crawler** de IBM. Se incluye en Analytics for Apache Hadoop, que es la versión en la nube de la máquina virtual ofrecida por IBM (IBM Infosphere BigInsights for Hadoop 3.0). Sencillo de utilizar. Sin embargo, esa sencillez implica que no se pueda realizar una ejecución realmente personalizada.

- **Httrack** [4]**.** Es un programa (software libre) multiplataforma, desarrollado en C y que posee una interfaz gráfica para su ejecución. Ofrece numerosos parámetros de configuración, por lo que se puede ajustar la búsqueda a los requerimientos de un trabajo concreto. Existen diversas opciones de prioridades para los enlaces: priorizar los archivos HTML o los que no lo sean, priorizar páginas del mismo dominio, directorio, subdirectorios o directorios superiores. Además, se puede elegir el número de procesos (enlaces) que se ejecutan (rastrean) en paralelo. Debido a que guarda una copia en local de los archivos que rastrea (útil para posteriormente tratar los datos de forma offline) se considera un crawler adecuado para ejecuciones en modo centrado, ya que resultaría inviable descargar en un equipo local todos los archivos de una búsqueda con un nivel de profundidad elevado y sin filtros.

- GNU **Wget** [5]. Además de ser un comando de consola con numerosos parámetros, se puede ejecutar en un script de forma que posteriormente se puedan manejar otros programas sobre los datos que el comando ha descargado de Internet. Está desarrollado en C, y es, además de multiplataforma, software libre. Posibilita la opción de descargar

recursivamente enlaces de una página web, hasta descargar la página web al completo. De manera similar al Httrack, guarda los archivos en local, y es más complejo de utilizar que el Web crawler de IBM pero más personalizable.

- Apache **Nutch** [6]. Está desarrollado en Java (por lo que es multiplataforma) y maneja Lucene como procedimiento para guardar los resultados. Es software libre. Su principal ventaja es la escalabilidad (dinámicamente escalable) y es apto para el modo de ejecución distribuido y en paralelo.

La Tabla 1 resume las características principales de los crawlers analizados.

Crawler	Descripción
Nutch	Buena escalabilidad Resultados con Lucene
Httrack	Resultados en local Focused crawling
Wget	Comando consola Resultados en local Muy personalizable
Web Crawler IBM en Bluemix	Sencillo Poco personalizable Computación en la nube
GoogleBot	Crawler de Google Distribuido y paralelo Algoritmo PageRank
BingBot	Crawler de Bing Distribuido y paralelo

Tabla 1. Crawlers libres y privados

Aunque el Web Crawler de IBM no sea software libre, ha sido posible manejarlo de forma online mediante la plataforma Bluemix [7], que es la PaaS (Platform as a service) de IBM. Eso significa que es una plataforma que ofrece diversos servicios para ser utilizados directamente por el usuario desde un navegador web mediante la interfaz gráfica proporcionada por la plataforma. Por lo tanto, el usuario no necesita hardware ni la instalación y configuración de software para la ejecución de los programas. Únicamente ha de disponer de un equipo con el que conectarse a Bluemix mediante un navegador. La plataforma se ejecuta sobre la infraestructura SoftLayer, que proporciona las capas inferiores a la plataforma y que también es de IBM. Bluemix está basada en la Cloud Foundry, que es una plataforma como servicio de código abierto. Para manejar Bluemix, el primer paso es la creación de una cuenta en la plataforma. Una vez que se posee la cuenta ya se pueden crear aplicaciones, que pueden ser desarrolladas en diversos lenguajes, como Python o Java. Posteriormente, se pueden asignar servicios a esas aplicaciones. Existe un

repositorio de servicios en EEUU y otro en Reino Unido (ambos están disponibles para su utilización). Dos de los servicios que se pueden emplear son:

- Analytics for Apache Hadoop. En el momento del análisis, este servicio equivalía a la máquina virtual "IBM Infosphere BigInsights for Hadoop 3.0". En ella se podían desplegar diversas aplicaciones, por ejemplo el Web Crawler. Sin embargo, este servicio ha sido actualizado con la versión 4.0 de dicha máquina virtual.

- Servicios de Watson. Son servicios de computación cognitiva, término que se describe en el siguiente apartado.

2.2 Text Analytics y Data Mining

Text analytics es un término que se refiere a los algoritmos que se emplean para obtener información estructurada de documentos que no presentan estructura, por lo que no se les pueden realizar consultas tradicionales. El término data mining, en cambio, engloba los algoritmos que obtienen información a partir de documentos que sí presentan una cierta estructura que el algoritmo pueda identificar. Además, existe otro término, denominado computación cognitiva, que engloba a los algoritmos que emulan el procedimiento humano de adquirir y emplear conocimientos.

2.2.1 Computación cognitiva

En la computación cognitiva se utilizan algoritmos tanto de data mining como de text analytics. El objetivo es que estos algoritmos sigan el procedimiento completo del razonamiento humano, para que puedan solucionar un problema completo sin la intervención de una persona. Watson [8] es un sistema de inteligencia artificial, creado por IBM, que proporciona diversos servicios cognitivos, tales como el servicio de preguntas y respuestas. Este servicio se incluye en computación cognitiva ya que imita el protocolo tradicional humano de: oír la pregunta, procesarla, hallar una respuesta basada en el conocimiento adquirido anteriormente por la persona y presentar la respuesta. Bluemix ofrece diversas demos de servicios de Watson:

- Concept Expansion: correlaciona los eufemismos o términos coloquiales con frases que se entienden comúnmente.

- Tradeoff Analytics: facilita a los usuarios la toma de decisiones.

- Concept Insights: explora información basada en conceptos.

- Question and Answer: al escribir una pregunta, el programa genera diversas respuestas con la probabilidad que tiene cada una de ser la más acertada. Al ser una demo, las preguntas tienen que entrar en el ámbito de los viajes o de la salud.

- AlchemyAPI: aunque esta aplicación, esté incluida en Bluemix como una demo de servicio de Watson, se puede descargar el SDK (Software Development Kit) en diversos lenguajes (C, Java, Python, Ruby, Php…) [9]. Una vez descargado, se pueden cambiar los archivos que se analizan por defecto, con el objetivo de aplicar los algoritmos a los documentos que se pretenden analizar. De este modo, se pueden contemplar resultados más personalizados que los ofrecidos por las demos. Funciona exclusivamente si los archivos que recibe poseen el formato HTML. Ejemplos de los diversos servicios que proporciona AlchemyAPI son:

    - Extracción de entidades: identifica entidades (personas, organizaciones, ciudades…) en un fichero HTML.

    - Análisis de sentimientos: indica si el texto posee una connotación positiva o negativa. Si la salida contiene la palabra "mixed", significa que el texto posee ambas connotaciones. No obstante, indica una puntuación entre -1 y 1, que evidencia el nivel de positividad general del texto.

    - Extracción de texto: este servicio suprime etiquetas HTML para extraer de una página web el texto plano que ésta contenga.

    - Extracción de categorías: indica la categoría a la que pertenece el archivo.

    - Etiquetado.

    - Extracción de palabras clave.

2.2.2 Topic Modeling

El término Topic Modeling [10] describe a los algoritmos (incluidos en la categoría de text analytics) cuya funcionalidad consiste en la obtención, a través de un modelo estadístico, de los temas sobre los que tratan diversos textos (corpus), que se analizan de forma conjunta. Cada grupo de palabras, que ocurren en diversas ocasiones de forma conjunta y que el algoritmo considera que pertenecen a un mismo tema, se denomina topic. Tras hallar los topics y la importancia de cada uno en el corpus, se indica el porcentaje que tiene cada uno en cada documento, con lo que se puede determinar la relación entre los textos.

Existen diversos algoritmos para realizar Topic Modeling. Uno de los primeros fue el PLSI (Probabilistic Latent Semantic Indexing), que fue creado por en 1999. Posteriormente, David Blei, Andrew Ng y Michel I. Jordan desarrollaron el algoritmo LDA (latent

Dirichlet allocation) [11]. Dicho algoritmo consiste en la generalización del PLSI empleando una distribución a priori de Dirichlet [12], la cual consiste en la generalización multivariada de la distribución beta.

Se han desarrollado diversos algoritmos que extienden el LDA, por ejemplo el algoritmo Pachinko allocation que, además de emplear las correlaciones entre palabras para crear los topics, maneja las correlaciones entre topics. No obstante, LDA es el método más extendido actualmente, ya que es el algoritmo que emplean por defecto diversos programas (que se analizarán posteriormente) que realizan topic modeling. Su funcionamiento se puede resumir con el esquema que aparece en la Figura 2.

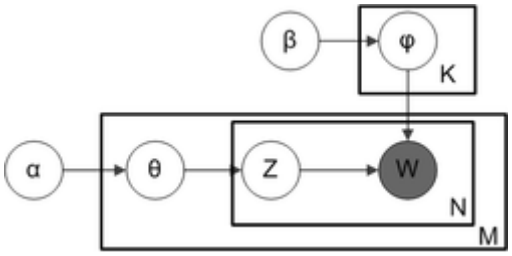

Figura 2. Esquema de funcionamiento del LDA

Los símbolos que aparecen en la Figura 2 poseen el siguiente significado:

- M corresponde al número de documentos, K al número de topics y N al número de palabras por documento.
- W es el conjunto de todas las palabras de la colección de documentos. Éste es el único parámetro observable. Todos los demás son latentes, por lo que son inferidos a partir de las palabras. W_{ij} representa a cada palabra i en el documento j.
- α es el hiperparámetro (debido a que es un parámetro de una distribución a priori) de Dirichlet de cada topic en un documento.
- β es el hiperparámetro de Dirichlet de cada palabra en un topic.
- θ_i es la distribución de topics para cada documento i.
- φ_k es la distribución de palabras para cada topic k.
- Z_{ij} indica el topic de la palabra j en el documento i.

Las variables aleatorias que surgen al desarrollar el esquema de la Figura 2 se pueden describir matemáticamente tal y como se observa en la Figura 3.

$$
\begin{aligned}
\boldsymbol{\varphi}_{k=1\ldots K} &\sim \text{Dirichlet}_V(\boldsymbol{\beta}) \\
\boldsymbol{\theta}_{d=1\ldots M} &\sim \text{Dirichlet}_K(\boldsymbol{\alpha}) \\
z_{d=1\ldots M, w=1\ldots N_d} &\sim \text{Categorical}_K(\boldsymbol{\theta}_d) \\
w_{d=1\ldots M, w=1\ldots N_d} &\sim \text{Categorical}_V(\boldsymbol{\varphi}_{z_{dw}})
\end{aligned}
$$

Figura 3. Descripción de las variables aleatorias del LDA

La Figura 3 se describe de la siguiente forma:

- $\varphi_{k=1\ldots k}$ indica la distribución de palabras en cada topic k. Se describe con la distribución de Dirichlet de β, la cual indicará finalmente la importancia de cada palabra en el topic.
- $\theta_{d=1\ldots M}$ indica la distribución de topics en cada documento d. Se describe con la distribución de Dirichlet de α, la cual indicará finalmente la importancia de cada topic en el corpus.
- $z_{d=1\ldots M,w=1\ldots Nd}$ indica el topic de la palabra w en el documento d.
- $w_{d=1\ldots M,w=1\ldots Nd}$ identifica la palabra w en el documento d.

Aprender estas cuatro distribuciones es un problema de inferencia Bayesiana. Un posible método para solucionar tal problema es el muestreo de Gibbs, que es un algoritmo MCMC ("Markov chain Monte Carmelo"). Por tanto, genera cadenas de Markov de las muestras que selecciona, y de esta forma, se relacionan las muestras más cercanas. De este modo, se favorece que palabras próximas en el texto posean más probabilidad de pertenecer al mismo topic. Este algoritmo es probabilista, por lo que hay que emplear una semilla en el programa para que los resultados coincidan en diferentes ejecuciones con los mismos parámetros.

Para ejecutar algoritmos de topic modeling se han empleado dos de los programas libres más extendidos para este fin y cuyas características se indican a continuación.

Mallet (MAchine Learning for LanguagE Toolkit) [13] es una colección de paquetes Java (desarrollada por Andrew McCallum, en colaboración con la universidad Massachusetts Amherst) orientada al procesamiento de textos mediante algoritmos de aprendizaje automático. Es uno de los estándares para la realización de topic modeling [14]. Al estar programado en Java, puede ejecutarse en diversos sistemas operativos. En Linux se puede ejecutar, una vez descargado, desde la consola. Para su utilización primero hay que ejecutar un comando con el cual se indica al programa dónde están los textos a analizar. Posteriormente, hay que ejecutar otro comando en el que se le indique el número de topics a emplear.

El **Stanford Topic Modeling Toolbox** [15] es un conjunto de herramientas que permite desarrollar programas de topic modeling mediante scripts en Scala. Al descargar el programa también se pueden incluir scripts de ejemplo para probar el funcionamiento y contemplar los resultados. Al funcionar mediante scripts es personalizable y potente.

Además de estos dos programas para realizar topic modeling, también existen librerías en **Python** con dicha funcionalidad (gensim y lda).

2.2.3 Clustering

Se define con el término de clustering a los algoritmos (incluidos en la categoría de algoritmos de data mining de aprendizaje no supervisado) que realizan procedimientos para agrupar observaciones, de forma que las que pertenezcan al mismo grupo sean más cercanas que las que no. La cercanía se mide con una función de distancia según los atributos (cada atributo se considera una dimensión) de las observaciones. Existen dos modos de realizar clustering: jerárquico y no jerárquico. Al igual que múltiples métodos de aprendizaje automático [16], ambos modos de clustering se pueden ejecutar con la librería scikit-learn de Python.

Modo no jerárquico. El número de clústeres se determina por el usuario antes de la ejecución del programa. A partir de los datos se hallan los centroides (valores de las dimensiones que posee el centro de cada clúster), y posteriormente se asigna cada observación a un clúster. Un ejemplo de este modo es el K-means. Para la ejecución de dicho algoritmo hay que especificar al programa:

- Inicialización de los centroides: se pueden inicializar de diversas maneras. En la librería scikit-learn de Python existen diversos métodos para realizar clustering [17]. En la función que aplica el algoritmo K-means, se permiten las opciones de inicialización de kmeans++ [18], de forma aleatoria o de la manera que requiera el usuario si introduce su propio algoritmo de inicialización. Se ha comprobado [19] que los métodos aleatorio y kmeans++ llegan a la misma solución en términos de inercia (suma de las distancias de cada punto con el centroide más cercano), pero el kmeans++ necesita menos iteraciones para obtener dicha solución, por lo que si se ejecuta el algoritmo con pocas iteraciones para la inicialización, resulta conveniente el kmeans++ (es el predeterminado en la función KMeans de scikit-learn).

- Número de clústeres: al ser no jerárquico, ha de indicarse el número de clústeres que se pretenden hallar. Si se requiere una clasificación general, conviene indicar un número reducido de clústeres para que únicamente distinga entre clústeres por las características más generales de los atributos. Al aumentar el número, el algoritmo cada vez escogerá características más individualizadas para construir cada clúster.

El resultado del clustering no jerárquico es un array en el que se indica a qué clúster pertenece cada punto. En la función de scikit-learn, además, se devuelven las coordenadas de los centroides y la inercia. Cuanto menor sea la inercia, más acertados serán los centroides. Además, cuanto mayor sea el número de clústeres, menor será la inercia, ya que habrá un mayor número de centroides, los cuales serán más personalizados (cercanos) a los puntos.

Modo jerárquico: el número de clústeres no se determina antes de la ejecución. Estos algoritmos pueden ser aglomerativos o divisivos. En el caso de los aglomerativos, la

ejecución comienza sin clústeres, y cada observación forma un clúster con la más cercana. Posteriormente, se forma un clúster que agrupe a los sub-clústeres más cercanos. Se realiza sucesivamente hasta que están agrupados todos los puntos en un solo clúster. De forma opuesta, un algoritmo divisivo comienza con todos los puntos en el mismo clúster, y en función de la lejanía de los datos, se divide en sub-clústeres. Un ejemplo de agrupamiento jerárquico es el método AgglomerativeClustering presente en la librería scikit-learn de Python. Para ejecutar este método hay que indicar las siguientes funciones:

- **Función de distancia**: función con la que se halla la distancia entre puntos y por tanto, con la que se decide qué grupos crear, ya que éstos se formarán a partir de la cercanía de los puntos. Existen múltiples funciones de distancia. En la función de scikit-learn se pueden utilizar: euclidiana (distancia ordinaria), manhattan (la distancia entre dos puntos es la suma de las diferencias absolutas de sus coordenadas) o coseno. La euclidiana es la función por defecto.

- **Función de enlace**: es la función que indica cómo se halla la distancia entre dos clústeres. Existen diversas opciones:

    - *Simple*: la distancia entre un clúster A y uno B es la distancia mínima entre cualquier punto del clúster A con cualquier punto del clúster B.

    - *Completa*: la distancia entre un clúster A y uno B es la distancia máxima entre cualquier punto del clúster A con cualquier punto del clúster B.

    - *Promedio*: la distancia entre un clúster A y uno B es la media de las distancias entre los puntos del clúster A con los puntos del clúster B.

    - *Ward*: es un algoritmo que elige la distancia que minimice la varianza. Debido a tal característica, usualmente es el que mejor resultados obtendrá. Es el predeterminado en scikit-learn.

La librería SciPy de Python, posee diversas funciones para realizar clustering jerárquico [20]. Con la función linkage se puede aplicar clustering jerárquico al conjunto de datos (dataset) elegido para ello. El resultado es una matriz con la información jerárquica que, posteriormente, se puede emplear con otra función de SciPy para crear un dendrograma [21]. Un dendrograma es una visualización de los resultados en la que se pueden contemplar los clústeres que existen en los distintos niveles de profundidad, así como la distancia entre ellos.

3 Desarrollo

En este apartado se indicará cómo se han aplicado a este trabajo, las diversas técnicas anteriormente explicadas. Primero se explicara cómo y con qué objetivo se han aplicado las técnicas de crawling. Posteriormente se indicarán los pasos realizados para pre-procesar y analizar los documentos con diversos métodos de text analytics. Finalmente se explicará cómo se han aplicado las técnicas de clustering a los resultados obtenidos mediante los métodos de text analytics.

3.1 Web crawling

La primera etapa del proyecto consistió en la recolección de discursos de rectores de diversas universidades. Debido a que únicamente se pretendían encontrar discursos universitarios, el modo de ejecución empleado en los crawlers ha sido el centrado, que se ha aplicado mediante el manejo de filtros.

Para la realización del trabajo se ha empleado, además del paradigma tradicional de ejecución de programas en local, el de computación en la nube. Esto se ha realizado tanto para la búsqueda como para el análisis de los documentos. Para la ejecución de programas en la nube se ha manejado la plataforma Bluemix. Tras la creación de una cuenta en la plataforma se creó una aplicación a la que se añadieron diversos servicios, para comprobar cuál de ellos incluía funcionalidades útiles para el trabajo. El servicio empleado para realizar web crawling ha sido "Analytics for Apache Hadoop". En dicho servicio se pueden realizar diversas tareas tales como subir ficheros del ordenador en el que se esté utilizando, ejecutar sobre ellos diversas aplicaciones y visualizar los resultados en forma de BigSheets, que son hojas de cálculo para Big Data. Una de las aplicaciones que se puede desplegar sobre este servicio es el IBM Web Crawler. También se puede desplegar otra aplicación de web crawling denominada BoardReader. Sin embargo, para utilizar dicha aplicación es necesaria una clave de una compañía externa a IBM.

Para realizar el trabajo se evaluaron distintas herramientas de rastreo. Las dos primeras fueron descartadas por los motivos que se exponen a continuación:

- Wget: ha sido empleado desde la bash en Ubuntu. Se ha utilizado para descargar una página recursivamente, ya que se sabía que en esa página se alojaban discursos. Sin embargo, no se ha empleado para el trabajo ya que, aunque sus características son parecidas a las de Httrack, con este último se ha podido desarrollar una búsqueda más ajustada a las necesidades del trabajo, debido al manejo de sus diversos parámetros.

- Apache Nutch: se utilizaría en un proyecto con un tamaño de datos mayor que el de los discursos debido a su escalabilidad, a su conexión con Hadoop y a que

la búsqueda de la información obtenida se realiza con rapidez debido al indexado que utiliza Lucene. No obstante, en este proyecto, el tamaño esperado de los discursos no era muy grande, por lo que los resultados se pretendían guardar en local para su posterior análisis.

Las otras dos herramientas sí que podían ser empleadas para la recolección de discursos y se analizaron en detalle:

- Web crawler de IBM: debido a que sigue el paradigma de computación en la nube, está diseñado de forma que su utilización resulte sencilla para un gran número de usuarios, incluso sin poseer grandes conocimientos sobre crawlers. Por lo tanto, el número de parámetros de la ejecución es el mínimo (enlaces con los que comenzar la ejecución, filtros, número máximo de profundidad permitida y número máximo de enlaces por cada nivel de profundidad). Sin embargo, esa sencillez implica el hecho de que no se pueda realizar una ejecución ajustada a las necesidades de la búsqueda de un tema concreto. Respecto a las políticas seguidas por el crawler:

    - No permite la elección de prioridades.

    - Cada filtro se expone por separado, lo cual impide indicar en qué parte del enlace han de aparecer los filtros. Esto también implica que no se puede utilizar el comodín '*'. Al realizar búsquedas automáticas con el filtro "+rector" y "+discurso", el programa aceptaría cualquier enlace de la página web "rectoria.com", lo cual no es lo que se espera. Al exponer ambos filtros con +, está implícito que todos los demás enlaces serán descartados (como si existiera un "-*").

    - Si se posee una cuenta en la que se permite ejecutar programas en diversos nodos, el crawler se paralelizará con una política interna.

- Httrack: guarda una copia en local de los archivos que rastrea, lo cual es lo que se necesitaba, ya que los discursos tenían que estar guardados para su posterior análisis (con excepción de los procesos de análisis en la nube). Respecto a las políticas que emplea:

    - Presenta la opción de manejar prioridades. No obstante, como ya se habían empleado filtros, los enlaces que pasaban dichos filtros debían tener la misma prioridad de ser rastreados, por lo que no se utilizó dicha funcionalidad. Debido a que no se establecieron prioridades, el manejo de filtros obtuvo más importancia, ya que si no se empleaban bien, el número de enlaces podría haber crecido demasiado, y al no haber

prioridades entre dichos enlaces, la posibilidad de rastrear un discurso disminuiría considerablemente.

- Permite la utilización de '*' en los filtros, por lo que se pueden crear filtros avanzados para indicar dónde debe el enlace contener dicho filtro. Por ejemplo, podemos indicar en un solo filtro que se descarguen solo los archivos PDF que contengan la palabra rector ("*rector*.pdf"). Además, se puede crear un filtro "*.*/*rector*.pdf" (para descartar el nombre de la web como lugar donde pueda contenerse un filtro). Con tal filtro, no se descargaría el enlace "www.rectoria.com/profesores", pero sí se descargaría el enlace "www.uam.es/discursosdelrector", lo cual es lo que se espera del crawler. Los filtros se indican por orden creciente de preferencia. Por lo tanto, el primer filtro ha de ser "-*", para indicar al crawler que descarte todos los enlaces. Como los siguientes filtros tienen preferencia al primero, al escribir "+rector", descartaría todos los enlaces, excepto los que contengan la palabra rector. Este crawler también posee otras opciones para cancelar la descarga de un enlace, tales como el tiempo máximo de respuesta o el tamaño máximo del archivo a descargar.

- Se puede elegir el número de procesos que se ejecutan simultáneamente (que sea realmente paralelo depende del número de procesos elegidos y de la cantidad de núcleos que se dispongan en el equipo en el que se está utilizando). No se empleó realmente una política de paralelización, ya que, aunque se ha ejecutado con la opción de rastrear ocho enlaces simultáneamente, al estar en el mismo PC, tenían el mismo proceso de frontera por lo que un proceso no podía rastrear un enlace que ya hubiera sido rastreado por otro.

- No se ha empleado política de volver a rastrear, ya que, aunque los discursos puedan ser actualizados o eliminados de las páginas web, la frecuencia de estos cambios es baja, por lo que no se ha requerido el volver a rastrear en el periodo que ha durado el trabajo.

Para la elección del programa a manejar para obtener los discursos que serán posteriormente analizados, es necesario comparar las ventajas e inconvenientes de ambos crawlers. Estas diferencias son las esperadas entre un servicio en la nube y un programa local. En primer lugar se mostrarán las ventajas del Web Crawler:

- Debido a que sigue el paradigma de computación en la nube, es una aplicación diseñada para una sencilla ejecución.

- Es posible que no se pueda disponer de un equipo encendido el tiempo suficiente hasta que finalice la ejecución del programa.

- Es probable que el clúster online disponga de más capacidad de cómputo que el equipo local.

No obstante, también posee una serie de desventajas, que se citan a continuación:

- Dependencia del servidor que aloje el servicio. Es posible que el servidor esté colapsado y apenas pueda ejecutar la aplicación. También es posible que, aunque no esté colapsado, exista un protocolo de prioridades entre aplicaciones y la que se quiera ejecutar no disponga de prioridad. Al utilizar el Web Crawler ha ocurrido en varias ocasiones que la página web no respondía a las peticiones del usuario, o tardaba considerablemente en hacerlo. Por tanto, la interacción es más rápida en local. Además, al realizar una ejecución sin filtros en la nube, la aplicación puede ejecutarse hasta que necesite demasiada capacidad de cómputo y sea cancelada por el servidor.

- Aunque el clúster en la nube tenga más capacidad de cómputo que el equipo local, debido a que con la cuenta básica de Bluemix las aplicaciones se ejecutan en un nodo, el programa local ha finalizado con más rapidez en diversas pruebas, ya que en la nube necesita un tiempo para inicializarse y además puede ser pausada o cancelada por el servidor.

- Actualización o eliminación de servicios en la página web. Esto ha ocurrido con "Analytics for Apache Hadoop" en Bluemix, que ha sido actualizado con la nueva versión de la máquina virtual "IBM Infosphere BigInsights for Hadoop 4.0". Dicha actualización simplifica la utilización de diversas utilidades. Sin embargo, suprime el despliegue de ciertas aplicaciones sobre el servicio, por lo que ya no se podría manejar el Web Crawler en Bluemix.

- No se obtienen los documentos para su manejo offline. Existen diversas opciones para manejar estadísticas sobre los resultados obtenidos. No obstante, no se puede analizar los textos completos.

Por tanto, se eligió Httrack como programa para obtener discursos debido a que se pueden ajustar los parámetros de la búsqueda y a que descarga los resultados para el manejo offline de los mismos. Una vez decidido el crawler a utilizar, para comparar resultados, se emplearon tres métodos para recolectar documentos mediante la ejecución de dicho crawler: manual, automático y semiautomático.

Método manual: se necesitan conocer los enlaces en los que se encuentran los documentos. Se pasa un enlace al crawler y se coloca el filtro que permita la descarga de los discursos. Resulta claro que este método será el que mejor porcentaje obtenga de

documentos encontrados respecto de los archivos que no lo sean. Sin embargo, este método no resulta práctico para el proceso total, ya que realmente solo está automatizando la descarga de los documentos, pero la búsqueda de los enlaces donde se encuentran sigue siendo una fase que ha de realizar el usuario previamente. No obstante, ha sido el método empleado en diversas universidades, en las cuales se conocían los enlaces y sus características y se requería la descarga de sus discursos.

Método automático: de manera opuesta al método anterior, se ha empleado un método en el que la obtención de documentos se realiza de una forma completamente automática. Previamente a la ejecución, se indica un enlace de una página web, por ejemplo de una que contenga los enlaces de todas las universidades a nivel mundial. A partir de dicho enlace, el crawler obtiene más enlaces con diferente nivel de profundidad para encontrar discursos. Este método sería el adecuado, ya que el crawler automatizaría todo el proceso. No obstante, en la práctica (con los recursos de los que se dispone) no resulta viable, ya que si se emplean filtros para descartar las páginas no deseadas, también se eliminarán las necesarias para llegar a los discursos. Por tanto, para llegar a los discursos habría que ejecutar el crawler sin filtros HTML (sí que podrían descartarse formatos de imágenes, audio o vídeo). Sin embargo, al ejecutar el programa sin filtros HTML, en cada enlace analizado se buscarán todos los hiperenlaces para añadirlos a la lista de enlaces pendientes de análisis. Por tanto, el número de enlaces crecerá exponencialmente. Cuanto más crece el número de enlaces en este método, más baja la probabilidad de éxito, es decir, de que cada archivo descargado sea un discurso de un rector. Tras realizar diversas pruebas, la tasa de acierto ha sido demasiado baja para ser considerada una de las opciones de rastreo.

Para encontrar enlaces base adecuados, se pueden emplear motores de búsqueda. Se ha probado con varios: Google, Duckduckgo, Yahoo, Bing, y otros que en realidad emplean la tecnología Google especializada en encontrar ciertos formatos. No obstante, la mayoría de crawlers no permiten que se empleen sus enlaces automáticamente. Por tanto, hay que introducir una etapa de descarga de los resultados de los buscadores para entregar dichos enlaces al crawler de forma local.

Si se busca en Google “discurso rector”, el buscador indica que se encuentran “aproximadamente 658.000 resultados”. Sin embargo, el buscador recoge los 1000 resultados que considere que poseen más importancia en dicha búsqueda y los muestra. Por tanto, serán esos 1000 enlaces los que se puedan crawlear. Para realizar la prueba, se escogió la opción de que aparecieran 100 resultados por página, y se descargaron las 10 páginas resultantes, por lo que ya se poseían los 1000 enlaces que muestra Google. Aparte del problema de la gran cantidad de enlaces encontrados que no eran discursos, al realizar esta forma de análisis, se han encontrado otros problemas, como que se puede hallar el mismo discurso a partir de varias fuentes, que se puede descargar un discurso sin conocer de qué universidad es (por lo que no se podrá indicar el ranking) y que se pueden descargar documentos de fuentes que no sean las propias universidades. Al descargar documentos de otras fuentes, aparte de que puedan ser discursos no verificados, es posible que se trate de

un artículo de prensa, el cual puede mezclar las palabras del rector con la opinión del periodista, lo cual introduciría datos no válidos para el análisis.

Al ejecutar el crawler con los 1000 resultados y con unos filtros que solo permitían archivos en formato PDF y enlaces que contuvieran la palabra rector o la palabra discurso, el número de resultados con un nivel máximo de profundidad igual a 3 fue de 3972. Con un nivel de profundidad de 4 fue de 13690. Aunque el número de discursos ha crecido considerablemente con solo un nivel más, si no se hubieran empleado filtros el crecimiento habría sido mayor, ya que una gran cantidad de los nuevos enlaces encontrados fueron descartados debido a los filtros.

Si se escoge como enlace base uno que contenga los enlaces de múltiples universidades (y se limita la profundidad máxima a otras páginas), en lugar de un buscador, se habrá solucionado el problema del reconocimiento de los discursos y de la fiabilidad de la fuente. No obstante, al no conocer en qué nivel de profundidad estarán los discursos dentro de la página de la universidad (ni la estructura que tendrán los enlaces para llegar a ese nivel), resulta complicado el manejo de parámetros como los filtros o el nivel de profundidad máximo.

Método semiautomático: debido a los problemas encontrados en los dos métodos anteriormente expuestos, se ha empleado el método que se ha denominado como semiautomático, el cual consta de dos fases. En la primera, se utiliza un buscador y se realizan consultas avanzadas de forma que los enlaces devueltos sean los enlaces en los que se encuentran los discursos. Posteriormente, en la segunda fase, se utiliza el crawler, pasándole como enlaces iniciales los que se han obtenido anteriormente. De esta manera, se puede ejecutar el programa sin filtros, ya que el nivel de profundidad máximo será un número reducido, debido a que se conocen los enlaces. Un ejemplo de búsqueda podría ser, buscar en Google "site:uam.es ext:pdf discurso rector" y, posteriormente, descargar todos los resultados como si ya fueran discursos. Como los discursos pueden estar en otro formato que no sea PDF y en esa ejecución se pueden descargar archivos que no sean discursos, es preferible realizar la ejecución en dos pasos. Primero se busca en Google "site:uam.es discurso rector", lo cual es muy posible que muestre como resultado, entre otras cosas, el enlace en el que se guardan los discursos. La mayor parte de las universidades que exponen sus discursos en Internet, lo hacen de una forma ordenada, por lo que encontrando la ruta base de los discursos, ya podrán ser descargados. Una vez que se obtiene tal enlace, se le indica al crawler, y se ejecuta con los parámetros adecuados para que descargue todos los enlaces requeridos.

Después de ejecutar el crawler, se pasa a una fase de validación de los resultados (si se han descargado manualmente los discursos de la web no es necesaria esta etapa), ya que existen numerosos archivos que se descargan pero que no son los documentos que se pretendía obtener. En general, cuanto más automático es el programa, mayor porcentaje de documentos no válidos encontrará. Si se emplean filtros, el número total de discursos hallados será menor, pero el porcentaje de documentos válidos será mayor.

La fase de eliminación de grano grueso de resultados inválidos se realizó de forma automática. Se emplea el término grano grueso para referirse a los archivos que son inválidos de una manera evidente, ya que poseen un formato que no puede ser el de un discurso de un rector. Dichos archivos fueron eliminados con un script de bash.

La validación continuó con la eliminación de grano fino, es decir, de textos que por el formato podrían haber sido un discurso, pero que no lo eran. Se podría haber continuado eliminando los archivos cuyo nombre no se asemejara al discurso de un rector (si no contuviera la palabra discurso ni la palabra rector). También se podría haber realizado un análisis de la estructura de los textos para determinar si eran discursos o no. Sin embargo, no todos los discursos poseen un título adecuado ni contienen la misma estructura de texto, por lo que, aunque el porcentaje de aciertos hubiera aumentado al realizar dichas acciones (eliminan más documentos inválidos que válidos), no se han realizado para no eliminar discursos válidos. Por lo tanto, la última etapa de la validación se ha realizado de forma manual.

A la hora de ejecutar el crawler se encontraron los siguientes problemas:

- Si se coloca “discurso” o “rector” como filtro del programa, la ejecución puede parar antes de llegar al enlace donde se encuentran los discursos. Es decir, si un discurso se encuentra en un nivel de profundidad 2 a partir del enlace base de la universidad (nivel 0) y el enlace que posee el crawler es el base, al colocar el filtro “rector”, posiblemente no entraría en el enlace de nivel de profundidad intermedio. Si no se colocan filtros, se descargará todo el contenido del enlace, y así sucesivamente con cada enlace, de forma que el número de enlaces en la frontera aumentará exponencialmente.

- El fichero robots.txt provoca que existan páginas en las que los discursos no puedan ser descargados de forma automática, ya que los crawlers que se han empleado respetan el código ético.

Los resultados se han obtenido mezclando el método semiautomático con el manual. De la página web “www.spellfinder.com/universities/” se ha obtenido el nombre y el enlace de múltiples universidades. En dicha página aparecen más de 8000 universidades, ordenadas por el país en el que se encuentran. Se comenzó buscando discursos en castellano, por lo que los enlaces rastreados fueron los de las universidades de España y Latinoamérica. Posteriormente, se han realizado búsquedas personalizadas para que los resultados del buscador únicamente incluyeran los enlaces de dichas páginas. De esta forma, un amplio porcentaje de los archivos descargados eran discursos. También se han descargado ciertos discursos con el método manual debido a la dificultad de emplear filtros para localizarlos. Esto es debido a que ciertas páginas no posicionan los discursos en una ruta y con un nombre adecuados.

En total, se han encontrado 3105 discursos, de los cuales 1640 estaban escritos en castellano. Como se ha realizado una búsqueda por todas las universidades de España, también se han encontrado diversos discursos en catalán, euskera y gallego. Los discursos restantes encontrados estaban en inglés o en alemán. 643 de los discursos tenían formato PDF, y, la mayor parte de los restantes, poseían el formato HTML. En realidad el discurso era una parte de dicho HTML, por lo que había que eliminar antes del análisis todas las partes de la página que no fueran el discurso.

3.2 Text Analytics

En este apartado se explicará el proceso seguido tras la obtención de los discursos. En primer lugar se han pre-procesado todos los documentos, para eliminar posibles complicaciones al algoritmo de análisis. Tras dicho pre-procesamiento, se han ejecutado algoritmos de text analytics sobre los discursos.

3.2.1 Pre-procesamiento

En el momento en el que se habían obtenido y validado los discursos, se añadieron los siguientes pasos que sirvieron de pre-procesamiento de los datos antes del análisis:

- Reconversión de todos los discursos al formato ".txt". Se ha elegido este formato ya que es texto plano, por lo que no contiene etiquetas que puedan perturbar el análisis del texto original. Para transformar los PDF a TXT se ha ejecutado un script de bash, empleando el comando pdftotext.

- Validar los textos, eliminando incongruencias debidas a una mala conversión entre formatos (sobre todo desde formato PDF).

- Elección del archivo con las palabras que se quieran descartar antes del análisis ("stopwords"). Ese descarte se realiza debido a que, usualmente, las palabras más frecuentes en los textos no aportan significado, ya que suelen ser preposiciones o artículos. Existen ficheros con estas palabras estándares para diversos lenguajes.

- Sustituir cada palabra por su raíz ("stemmizar"). Se realiza debido a que el LDA contabiliza el número de ocasiones que se repite cada palabra, y, por otro lado, la sintaxis de las palabras varía al añadirles prefijos y sufijos (los cuales usualmente no varían el significado de la palabra). Por tanto, si no se halla la raíz, el programa analizará palabras como por ejemplo "estudiante" y "estudiantes" como diferentes, y, es posible que analizadas de forma individual no obtengan una importancia considerable en el corpus, pero al analizarlas como la misma palabra obtengan una mayor importancia en un determinado topic. Si se stemmiza, los topics serán un conjunto de palabras que tengan relación en los textos, pero que no tienen por qué tener relación en general (es lo que se pretende al buscar relaciones entre textos).

Esto se ha realizado con un script en Python, empleando la función stem, de la librería stemming, con todos los discursos. Además, se ha aplicado a las stopwords, ya que si no se hiciera así, no descartaría estas palabras (para el posterior análisis del LDA) de la forma stemmizada en la que aparecerán en los discursos.

3.2.2 Topic Modeling

Una vez obtenidos, validados y pre-procesados los documentos, se pasa a analizarlos. La opción principal de análisis fue aplicar topic modeling. No obstante, también se han ejecutado otros métodos de text analytics, como los servicios presentes en Bluemix. Una ventaja de Bluemix es que diversos servicios se encuentran juntos en la misma web y se pueden probar fácil y rápidamente, para comprobar cuál es el que otorga una funcionalidad más parecida a la que se requiere. Por otro lado, las demos y las aplicaciones en fase beta son desventajas, ya que propician que no se pueda comprobar completamente si el producto es útil para un trabajo concreto. Se han probado diversos servicios Watson en Bluemix ("Concept Insights" y "Question and answer"). Sin embargo, no se han empleado en el trabajo, ya que, aparte de ser demos, la funcionalidad que poseen no cubre las necesidades planteadas como objetivo. El servicio cuya funcionalidad se asemeja más a los requisitos fue AlchemyAPI. Para la realización de pruebas en esta aplicación se descargó el SDK en Java, y se ejecutó desde la consola de Ubuntu. Debido a que esta aplicación solo acepta documentos con formato HTML, los archivos que habían sido previamente convertidos en TXT, tuvieron que reconvertirse a HTML. No obstante, continuaban siendo texto plano, ya que se desarrolló un programa que simplemente añadía las etiquetas básicas a los textos para ser reconocidos como HTML. Los servicios probados han sido:

- Extracción de entidades: podría resultar útil como pre-procesamiento, para descartar los nombres de personas, ciudades e instituciones, ya que éstos harían que los textos difirieran, pero no indicarían cual es el tema general.

- Análisis de sentimientos: se pueden ordenar los resultados y examinar en busca de patrones de positividad entre distintos discursos. Sin embargo, como era de esperar, tras realizar la ejecución con los discursos, no ha habido patrones destacables basándose en el sentimiento general del texto. La mayoría tenía un ranking positivo (entre 0,4 y 0,8).

- Extracción de texto: este servicio se ha empleado para extraer el texto de documentos HTML, y de esta forma aplicarlo en otros programas que acepten el formato TXT.

- Extracción de categorías: podría ser útil para una primera clasificación de discursos. Sin embargo, es una clasificación demasiado general para este trabajo. Además, no es una clasificación apropiada para diferenciar la calidad de diversas universidades,

ya que, incluso siendo de la misma universidad, es probable que diversos discursos estén incluidos en distintas categorías, ya que, pueden tratar de temas diferentes.

Por tanto, estas herramientas podrían resultar útiles para un determinado aspecto del trabajo, aunque con ellas no se puede realizar un análisis con una funcionalidad similar al topic modeling. Las aplicaciones de AlchemyAPI que más se asemejan son "Etiquetado" y "Extracción de palabras clave". Aunque ambas aplicaciones tienen funcionalidades semejantes, la diferencia radica en que se considera que una etiqueta describe el contenido de un texto de una manera clara para el usuario (aunque las palabras exactas de la etiqueta no aparezcan en el texto) y, en cambio, una palabra clave, es un término presente en el texto cuya importancia es significativa para un motor de búsqueda (los términos de ese texto más buscados por los usuarios en dicho motor de búsqueda).

- Etiquetado: es un servicio que obtiene etiquetas de textos. La limitación es que únicamente funciona con textos en inglés. Esta aplicación analiza y expone los resultados de forma individual para cada documento. Dicho resultado consiste en un grupo de etiquetas para cada texto, con la relevancia de cada una y los enlaces de las bases de datos de las cuales se ha obtenido la información para generar tales etiquetas.

- Extracción de palabras clave: extrae las palabras más importantes de los textos. El resultado es una serie de palabras ordenadas según la relevancia que poseen en ese determinado texto. Este servicio funciona en 8 idiomas (inglés, español, alemán, francés, italiano, ruso, portugués y sueco). Al igual que el etiquetado, funciona de manera individual para cada texto. El resultado es el conjunto formado por el nombre del documento, el idioma en el que está escrito y los diversos pares formados por cada palabra clave junto con la relevancia que posee.

Estas aplicaciones no han sido escogidas para el análisis final de los discursos debido a que funcionan de forma individual, y la base del topic modeling radica en analizar los discursos de forma conjunta para posteriormente buscar semejanzas. Esta funcionalidad no se ha encontrado en AlchemyAPI ni en los diversos servicios de Bluemix. Por tanto, para realizar topic modeling, se ha empleado Mallet, ya que es una herramienta completa y con un uso extendido para este fin, como se describe en la siguiente sección. El algoritmo de topic modeling ejecutado ha sido el LDA.

3.2.3 Mallet

Se instaló la versión 2.0.7 de Mallet en Ubuntu. Posteriormente se podía utilizar el programa desde la bash. Este programa cuenta con múltiples comandos, aunque se explicarán únicamente los dos necesarios para realizar topic modeling. Como se muestra en la Figura 4, en el primer comando se indican los ficheros que se van a analizar y el archivo de stopwords.

```
bin/mallet import-dir --input ../englishStemmed --output
../englishStemmed.mallet --keep-sequence --remove-stopwords
--stoplist-file ../stoplists/mixStemmed.txt
```

Figura 4. Comando importación datos en Mallet

El significado de cada parámetro de la Figura 4 es el siguiente:

- Import-dir --input: en Mallet se puede importar un fichero y que analice cada línea, o importar un directorio y que analice cada documento. Con este parámetro se importa un directorio entero, y es el que se ha empleado para importar todos los discursos una vez pre-procesados.

- Output: indica dónde se guarda el fichero ".mallet" que se utilizará como documento de entrada en los siguientes comandos.

- Keep-sequence: se emplea para indicar que los datos son una secuencia de palabras.

- Remove-stopwords: suprime las stopwords que se indiquen tras el parámetro "stoplist-file". Este programa contiene diversos archivos con stopwords por defecto, cada uno en su correspondiente lenguaje. No obstante, se ha creado un nuevo archivo, denominado "mixStemmed", debido a que se han mezclado tanto las palabras que son stopwords por defecto en inglés, como múltiples palabras que aparecían en los textos pero que no aportaban significado (fundamentalmente nombres de personas, ciudades e instituciones).

- Para las pruebas de Mallet en discursos en castellano hubo que añadir el comando "--token-regex '[\p{L}\p{M}]+'", ya que la versión predeterminada está orientada al inglés y no reconoce caracteres como las tildes o la 'ñ'.

- Mallet por defecto transforma todas las letras a minúscula, pero esto se puede evitar empleando el parámetro "--preserve-case"

Una vez se ha ejecutado el primer comando y se posee el archivo ".mallet", se ejecuta el segundo, el cual se observa en la Figura 5.

```
../bin/mallet train-topics –input ../englishStemmed.mallet --num-
topics $i --num-top-words 10 --output-doc-topics csv/topic$i.csv
--output-topic-keys english_keys$i.txt --optimize-interval 20
--optimize-burn-in 50 --random-seed 3 > /dev/null 2>&1
```

Figura 5. Comando Mallet de Topic modeling

A continuación se explican los parámetros observados en la Figura 5:

- Num-topics: indica el número de topics que debe encontrar el programa. El programa se ha ejecutado en un script de bash, por lo que el parámetro $i indica el número de topics de cada iteración. Se ha ido variando dicho parámetro entre 2 y 100 topics. Posteriormente, cuando se aplique el algoritmo de clustering, se visualizará qué número de topics resulta más adecuado.

- Num-top-words: indica el número de palabras que se muestran con cada topic. También ha ido variando en los scripts, aunque, finalmente, se ha establecido en 10 para que la diferencia entre diversas ejecuciones fuera solo el número de topics.

- Output-doc-topics: parámetro que indica dónde guardar el principal documento de salida del programa. En dicho documento (elegido con formato CSV), cada fila contiene un discurso y cada columna es el número de topic junto con el porcentaje que posee tal topic en dicho discurso (ordenados por la importancia de cada topic en cada documento). Este será el documento base para la clusterización.

- Los parámetros "optimize-interval" y "optimize-burn-in" son parámetros de optimización. El primero indica el número de iteraciones que han de transcurrir para que el programa recalcule el peso de cada topic en el corpus. El segundo indica el número de iteraciones iniciales hasta que el programa comienza a calcular el peso de los topics. Si no se emplean estos parámetros de optimización, LDA no mostrará la importancia de cada topic en el corpus. Por tanto, al emplearlo, se halla el parámetro α de Dirichlet y se obtienen mejores topics al conocer el peso de cada uno durante el proceso.

- El resto de archivos de salida de Mallet no han sido empleados para clusterizar, aunque resultan útiles para visualizar la importancia de cada topic en el corpus o de cada palabra en un topic. Por ejemplo, "output-topic-keys" muestra el número "num-top-words" de palabras para cada topic, junto con el parámetro de Dirichlet correspondiente. Debido a que se ha empleado el parámetro "optimize-interval", el número indicado para cada topic equivale a la importancia de dicho topic en el corpus.

- Random-seed: se utiliza para indicar la semilla que ha de emplear en cada iteración, para que los resultados no varíen entre diversas ejecuciones del programa con los mismos parámetros, y de esta forma poder comparar resultados al variar el número de topics.

- Num-iterations: indica el número de iteraciones que empleará el muestreo de Gibbs que maneja Mallet. Con un número de iteraciones bajo, el programa tarda poco tiempo en ejecutarse. No obstante, cuantas más iteraciones, mejor se calculará la diferencia de importancia entre diversas palabras, y por tanto, mejor se hallarán

tanto los topics como el peso de cada uno. En la Tabla 2 se muestra el tiempo de ejecución para distintos valores de este parámetro. Tras aplicar los métodos de clustering que se explicarán a continuación, se indicará cuál es el valor más adecuado.

Número iteraciones	Tiempo (segundos)
50	2,3
100	3,42
500	12,63
1000	25,71
3000	76,33
10000	245,4

Tabla 2. Tiempo de ejecución variando el número de iteraciones

3.3 Data mining: clustering

Una vez finalizadas las ejecuciones de Mallet, se aplica el algoritmo de clustering al archivo CSV obtenido. Se ha desarrollado un programa en Python con el propósito de transformar dicho documento, de forma que el resultado sea comprensible para un algoritmo de clustering. Dicho programa elimina las etiquetas y descripciones que había colocado Mallet, dejando en el documento tantas columnas como topics se hayan empleado. Además, se ordenan dichas columnas por el número de topic, en lugar de por la importancia que posee cada topic.

Posteriormente, se han empleado algoritmos de clustering en Python. En primer lugar, se ha utilizado la librería scikit-learn para ejecutar, tanto el algoritmo K-means, como el aglomerativo. Con estos algoritmos se ha obtenido una tabla que indica a que clúster pertenece cada discurso. Además, se han probado diversas funciones de distancia y de enlace para el algoritmo aglomerativo.

Como con las tablas devueltas por los algoritmos de scikit-learn no se podía apreciar la distancia entre puntos, se utilizó la función linkage de SciPy. Se ha ejecutado este algoritmo con diversas funciones de enlace (para la distancia se emplea la euclidiana). Las gráficas varían en cada ejecución al variar el número de topics. No obstante, aunque varíe el color y la organización de los clústeres, en ellas se aprecian diversos subgrupos en los que predominan discursos con el mismo ranking. Los resultados de clustering se explicarán en el siguiente capítulo.

4 Pruebas y resultados

Las pruebas de evaluación del trabajo comenzaron empleando tanto el modo de crawling semiautomático como el manual con el crawler Httrack para la obtención de discursos que cumplan las siguientes condiciones:

- Estar escritos en un mismo lenguaje. Esto se ha realizado para que el algoritmo de topic modeling no encontrara diferencias entre dos palabras que significan lo mismo pero que están escritas en un idioma distinto. Se ha elegido el idioma inglés ya que es el más extendido en el ámbito universitario.

- Estar incluidos en uno de los dos rangos diferenciados según la clasificación de la universidad en el ranking de Shanghái [22] (que clasifica a 500 universidades): el rango A que incluye universidades que aparecen en los puestos del 1 al 60 y el rango B que incluye a las que ocupan posiciones a partir de la 300 .De esta forma, comprobará si esta diferencia de ranking es apreciable por el análisis de los temas de los que tratan. En la Tabla 3 se puede visualizar la clasificación de las universidades de Ranking A y en la Tabla 4 las de Ranking B.

En total se han obtenido 276 discursos, 161 incluidos en el ranking A y 115 en el ranking B. Todas las universidades del ranking B son de EEUU, al igual que múltiples del ranking A. De esta forma, la diferencia de ranking no se puede atribuir a la localización geográfica de la universidad. No obstante, para que los topics no se centraran en posibles características de un mismo país, se han incluido en el ranking A universidades de otros países como la universidad de Oxford, la universidad de Kyoto o la universidad de Oslo). Las Tablas 3 y 3 también incluyen el número de documentos de cada universidad presentes en el corpus.

Universidad	**Ranking Shanghái**	**Discursos**
Harvard	1	10
Stanford	2	15
MIT	3	12
Berkeley	4	3
Cambridge	5	18
Princeton	6	27
Caltech	7	2
Columbia	8	15
Oxford	10	7
Imperial college London	23	8
Universidad Toronto	25	19
Kyoto	26	5
Uio	58	20

Tabla 3. Discursos clasificados con Ranking A

Universidad	Ranking Shanghái	Discursos
Uconn	300	8
Drexel	300	12
Georgetown	300	13
Oregon	300	10
Rensselaer	300	22
Temple	300	12
Alaska	400	6
Arkansas	400	6
Clemsom	400	7
Maryland	400	4
Medical Carolina	400	8
Syracuse	400	2
Kent	500	5

Tabla 4. Discursos clasificados con Ranking B

El siguiente paso fue ejecutar Mallet para aplicar topic modeling a dichos discursos. Para ello se han empleado los parámetros indicados en el capítulo de desarrollo. En la Figura 6 se observan los resultados tras ejecutar el programa empleando 10 topics.

```
0       0.50853 research world global fund educ support challeng develop inter
1       0.25741 scienc technolog scientif scientist global world engin energi
2       1.17834 public question understand institut requir human fact iss resp
3       0.19883 student templ educ percent teach learn higher art public
4       0.66369 world today life peopl work challeng moment servic honor
5       0.26274 today intern world global prize human iss peac university
6       0.15206 research award health medic care work system team receiv
7       0.15782 women polit person men leadership peopl age ethic run
8       1.88776 student educ university work academ research opportun support
9       0.26705 campus region work develop fund research wer communiti peopl
```

Figura 6. Topics hallados

Además del número que identifica cada topic y las palabras de las que se compone, se puede visualizar en la Figura 6 el parámetro α de la distribución de Dirichlet, el cual indica la importancia que posee cada topic en el corpus. También puede observarse que el topic 8 es el que mayor relevancia tiene. Esto es comprensible, dado que trata de temas generales para todos los discursos, tales como la universidad, los estudiantes y la educación. De la misma forma, se comprueba que el topic que presenta menos importancia en el corpus es el 6. También es comprensible ya que trata de premios, de cuidados médicos y de salud. Al ser un tema más especializado, está presente en menos discursos y, por tanto, posee una importancia menor en el corpus.

Es de esperar que si se analizan los discursos que poseen un porcentaje considerable de relevancia en topics con poca importancia en el corpus, serán discursos especializados en dicho tema. Esto se ha probado con el topic 1, el cual está relacionado con la ciencia y la tecnología. El discurso que más proporción posee de este topic es “2013predictions.txt”, de

la universidad de Rensselaer. Al leer el discurso, se comprueba que trata de las predicciones (Big Data e Internet of Things) para el futuro, realizadas por el rector de dicha universidad. Por lo tanto, se cumple que el texto es específico en el área de la tecnología. Lo mismo ocurre tras analizar el topic 6 (investigación, salud y medicina), y comprobar que el texto que posee más porcentaje de dicho topic es un discurso de la universidad de Maryland que recoge los logros que ha conseguido la universidad en dicha área.

Una vez realizadas las pruebas de topic modeling, se pasó a ejecutar algoritmos que pudieran cuantificar resultados a partir de los temas hallados. En primer lugar, se ejecutará un método de análisis supervisado (red neuronal), y posteriormente un método de aprendizaje no supervisado (clustering jerárquico y no jerárquico).

4.1 Perceptrón Multicapa

La primera de las pruebas de análisis consistió en aplicar una red neuronal artificial (perceptrón multicapa desarrollado durante una asignatura del grado) al fichero CSV que contiene la proporción de cada topic en cada discurso. El perceptrón multicapa es un método de aprendizaje supervisado, ya que consta de una etapa de entrenamiento, en la cual se le indica el valor correcto de las salidas junto con las entradas. Posteriormente, en la etapa de test, se comprueba si la red neuronal ha aprendido bien. Al ser multicapa, puede resolver problemas no separables linealmente. En esta prueba, el target "1 0" indica la salida de la red esperada para los discursos de ranking A y el "0 1" para los del B.

Se presupone que el perceptrón no podrá aprender el problema sin errores, ya que no es un problema completamente separable debido a que existen discursos con temas similares en diversas universidades. No obstante, se calculará la tasa de error que obtiene la red tras ser entrenada. Como existen más discursos en el ranking A, para realizar la prueba se han elegido aleatoriamente 115 discursos de dicho ranking para entrenar y probar la red junto con los 115 del ranking B. Si tras hacer la media en un script con múltiples ejecuciones con esos 230 discursos, la tasa de error no posee un valor cercano al 50%, significa que la red ha sido capaz de encontrar ciertas semejanzas en múltiples discursos con un mismo ranking. La Tabla 5 muestra el error en la época de test empleando 2, 4 y 5 topics. Como parámetros de ejecución del perceptrón se ha utilizado un porcentaje del 75% de los discursos como conjunto de entrenamiento, un porcentaje del 25% como conjunto de test y se han empleado 8 neuronas en la capa oculta.

Número de topics	Porcentaje error en test
2	39,03%
4	33,45%
5	30,18%

Tabla 5. Resultados del perceptrón multicapa

Puede observarse en la Tabla 5 que el error desciende al aumentar el número de topics, ya que al usar solo 2, los topics son demasiado generales para distinguir entre universidades. En cambio, con 5 topics el error baja al 30%. Dicho número, aunque no es un valor demasiado preciso, está alejado del 50% de fallos, que sería el valor esperado para predicciones al azar. Por tanto, existe, en los discursos con mismo ranking, cierta semejanza que hace que la red acierte 7 de cada 10 predicciones.

La siguiente prueba consiste en comprobar si la red puede discernir entre discursos de dos conjuntos de universidades más próximos en el ranking que en la prueba anterior. Para ello, se han seleccionado los discursos de las universidades con un ranking 1-4 (clase 1) y con un ranking 20-60 (clase 2). El error en esta prueba, usando 5 topics, ha sido de 44,77%. Este valor, menos adecuado que el obtenido en la prueba anterior, indica que cuanto más próximas son las universidades en ranking, más le cuesta a la red diferenciar entre sus discursos.

Por tanto, con la red neuronal se ha cuantificado la semejanza entre los discursos del mismo ranking. No obstante, con esta prueba no se han podido visualizar los discursos individuales que se asemejan. Para conocer esta información, se han realizado diversas pruebas con algoritmos de clustering que se describen en la siguiente sección.

4.2 Clustering

Para la prueba básica se han empleado dos clústeres (al igual que la red neuronal empleaba dos clases de salida). Aunque esta no es la forma habitual de utilización de clustering, sirve como prueba exploratoria para observar qué porcentaje de ambos rankings (A y B) se observan en cada clúster. Se ha empleado tanto el algoritmo K-means como el jerárquico de scikit-learn. El algoritmo K-means con inicialización aleatoria o con inicialización kmeans++ ha dado el mismo resultado (con solo dos clústeres), por lo que para realizar la prueba se empleó el método de inicialización kmeans++. Se ha variado el número de topics entre 4 y 50. Tras la ejecución se comprobó que el algoritmo no realizaba ninguna división que tuviera utilidad para diferenciar discursos con distinto ranking (A o B), ya que el porcentaje de cada clúster variaba considerablemente con cada ejecución.

Posteriormente se realizó la misma prueba con el algoritmo jerárquico, variando la función de enlace (completa, promedio y ward) y de distancia (euclidiana, manhattan y coseno). En la Tabla 6, se muestra el porcentaje de discursos que aparecen en el clúster 1, tanto para el ranking A (verde), como para el ranking B (naranja). La función Ward solo se ejecuta con la función euclidiana, ya que es un algoritmo pensado para optimizar la varianza con dicha función de distancia. En la Tabla 6 se observa que la mayor parte de los discursos están en el clúster 1, por lo que no realiza una clasificación de utilidad según el ranking.

	Euclidiana	Manhattan	Coseno
Completa	86% 88%	95% 89%	92% 88%
Promedio	100% 97%	100% 97%	96% 95%
Ward	72% 77%		

Tabla 6. Universidades con rango A-rango B

Es comprensible que la red neuronal obtenga más diferencias que el clustering con solo dos clústeres, ya que previamente ha sido entrenada para encontrar dichas diferencias. No obstante, en la Tabla 7, al aplicar 10 clústeres, se puede observar cierta diferencia en el número de discursos que pertenecen a cada clúster.

Ranking	Número de discursos en cada clúster									
A	32	28	5	30	14	3	30	7	11	1
B	18	7	6	22	28	2	11	6	9	6

Tabla 7. Discursos en cada uno de los 10 clústeres empleados

Sin embargo, no se conoce el número de clústeres óptimo para distinguir entre discursos del ranking A y discursos del ranking B. Además, aunque se hallara dicho número de clústeres óptimo, se seguiría sin visualizar la existencia de patrones, ya que no se visualizan las distancias entre cada discurso. Por ese motivo, la siguiente prueba ha consistido en emplear el clustering jerárquico con SciPy para crear un dendrograma donde visualizar los resultados.

El primer paso en dicha prueba ha sido ejecutar los diferentes métodos de enlace, y observar la diferencia en las gráficas. Como se obtuvieron resultados similares al variar la función que mide las distancias, en las siguientes gráficas se ha empleado la función euclidiana, que es la que se utiliza por defecto.

La Figura 7 muestra el dendrograma empleando 10 topics y la función Ward de enlace. También se ha ejecutado el algoritmo con las funciones simple, completa y promedio (Anexo A). Se ha escogido el método Ward debido a que, tal y como se observa en el anexo, es en la que mejor se visualiza la presencia de clústeres de discursos del mismo ranking. Además la elección del valor 10 para el número de topics a emplear y del valor 1000 para el número de iteraciones que realiza el muestreo de Gibbs, se ha realizado tras ejecutar con diversos parámetros y visualizar los resultados (Anexo B). En general, con menor número de topics, por ejemplo con 2, no se diferenciaba bien entre universidades, ya que los topics, al ser pocos, eran muy generales, y todos los discursos los contenían con proporciones considerables. Al emplear un mayor número de topics, por ejemplo 100,

existían topics demasiado específicos para cada discurso (además de los generales). De esta forma, casi todos los discursos poseían demasiada proporción de los topics generales en comparación con los individuales, por lo que al realizar el dendrograma aparecían con bastante proximidad. Además, al emplear un número alto de topics, los topics serán tan individualizados que, además de encontrar posibles diferencias entre discursos de universidades de con distinto ranking, pueden encontrar diferencias entre cualquier discurso, incluso siendo de la misma universidad. Esto no resulta útil ya que en este trabajo todos los discursos de la misma universidad tienen la misma clasificación (la que tenga dicha universidad en el ranking de Shanghái).

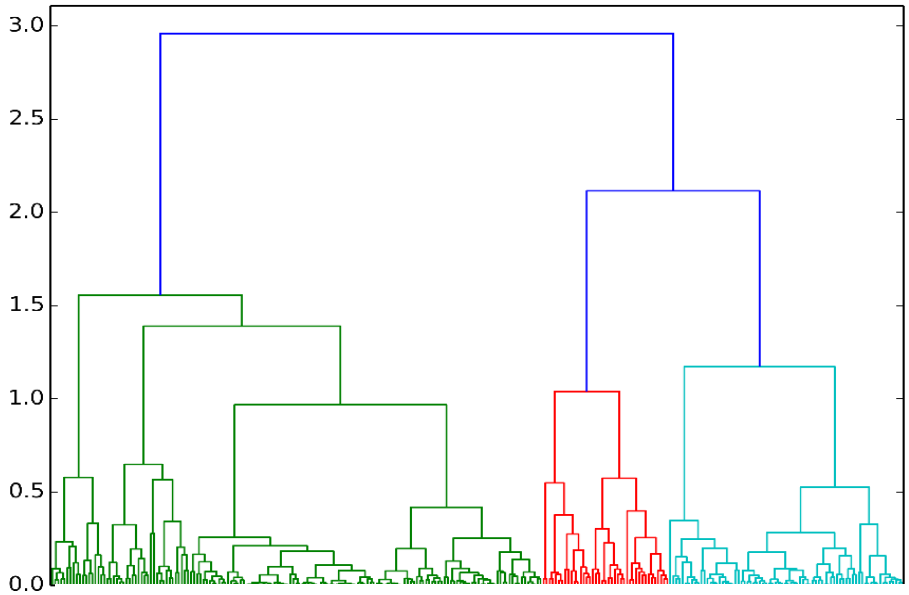

Figura 7. Dendrograma con 10 topics y función de enlace Ward

Los discursos (cada hoja del dendrograma) están etiquetados con el ranking que posee la universidad donde se han pronunciado, ya que, debido a que no importan las diferencias individuales entre discursos de la misma universidad. En la Figura 7 no se visualizan dichas etiquetas, pero se podrá comprobar tanto en la Figura 10 como en la 11. Observando la Figura 7, se comprueba que existen 3 colores (verde, rojo y azul) que realizan la separación general de los discursos en tres grupos (cada hoja del dendrograma tiene uno de esos tres colores). Los discursos con color azul corresponden al predominio del topic general, por lo que este grupo posee discursos en ambos rangos. En concreto, en las Figuras 8 y 9 se van a analizar dos de los discursos más próximos en el dendrograma.

125	Ranking A, Universidad de Toronto: “2014lgtb.txt”			
	8	0.641596877043699	5	0.17366471584339085
	2	0.10013010182198226	4	0.060896099850469516
	0	0.013097034500687792	9	0.006615273689070434
	1	0.0013439586680769925	3	0.001038076877942365

Figura 8. Topics de un discurso concreto (1)

156	Ranking A, Universidad de Stanford: “freedom.txt”			
	8	0.6402557198238238	4	0.18119412020225942
	2	0.12106484665075422	0	0.029155466062633126
	1	0.01653144969670782	7	0.008378771528813767
	9	0.0010369546718941235	5	0.0010202011604081307

Figura 9. Topics de un discurso concreto (2)

Se observa que los discursos cuyos topics aparecen en las Figuras 8 y 9 aparecen tan cercanos en el dendrograma debido a que tienen un valor muy similar en el topic 8, que es el más importante en ellos, por lo que estarán en un sitio cercano en esa dimensión, al igual que en la dimensión del topic 2. Leyendo los dos discursos se comprueba que ambos textos hablan sobre los estudiantes y las libertades que deben tener y se refieren a temas muy genéricos tales como estudiantes, oportunidades y comunidad. Por tanto, están cercanos en los topics que más importancia han tenido en el corpus. De ese modo, se puede concluir que la cercanía de los discursos que aparecen en color azul en la Figura 7 es debida a la ausencia de topics específicos que los diferencien.

Continuando con el análisis de la Figura 7, los discursos que aparecen en color verde sí poseen ciertas características que les diferencian. En concreto, en la zona izquierda del clúster predominan los discursos de universidades con ranking B, y en la zona derecha los que poseen ranking A. Esto puede apreciarse en la Figura 10, en las que se observa en detalle los clústeres donde predominan los discursos con ranking A. Posteriormente, en la Figura 11 se puede observar el clúster rojo de la Figura 7, en el cual predominan los discursos con ranking B.

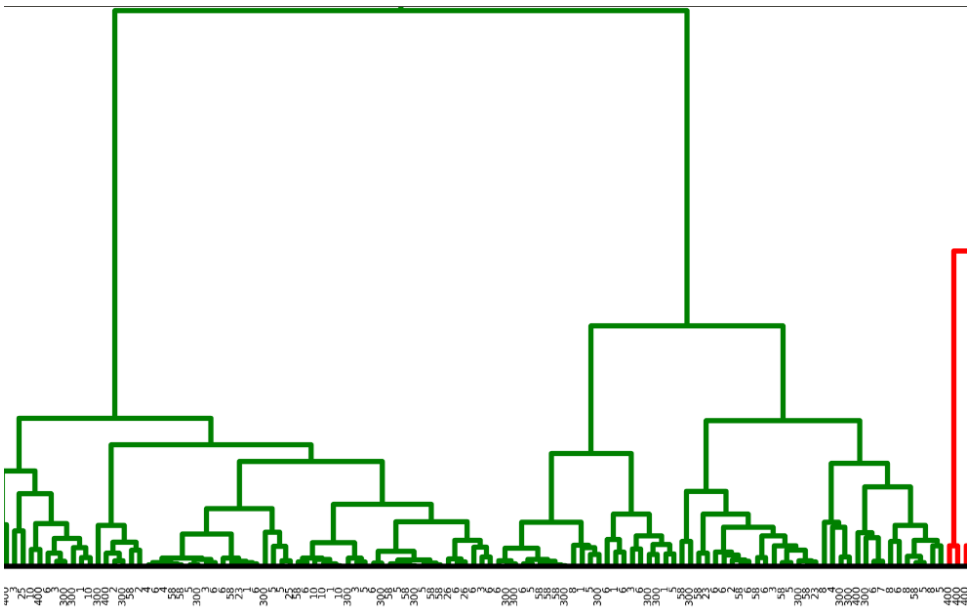

Figura 10. Clúster predominado por discursos de Ranking A

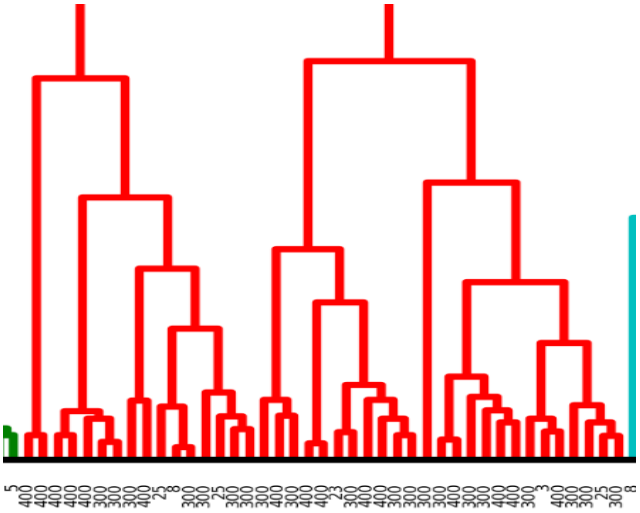

Figura 11. Ranking predominado por discursos de Ranking B

Este resultado es el más relevante ya que indica que existen similitudes entre los temas sobre los que tratan los discursos pronunciados en las universidades más prestigiosas y también entre las que ocupan los lugares más bajos en el ranking.

Para intentar analizar más detalles de este resultado, se analizarán los topics sobre los que tratan los discursos de cada uno de estos clústeres.

En los discursos de la Figura 10 predomina, además de los topics 2 y 8 que son los topics que hablan de temas comunes en todos los discursos (por lo que no aportan demasiada información del clúster concreto), el topic 4. El topic 5, que también está presente en el clúster, es un ejemplo de topic individualizado que no presenta importancia en el corpus. De acuerdo con la Figura 6, este topic posee un nivel de relevancia en el corpus de 0,26, debido, fundamentalmente, a la prevalencia de este tema en los discursos de la Universidad de Oslo, que tratan, en general, sobre humanidad, premios y el premio Nobel de la Paz. Por lo tanto, complica el buscar relaciones, aunque en este caso se han encontrado debido al topic 4.

Aunque en la Figura 11 se observa el predominio de los discursos de ranking B, es necesario que cada discurso tenga su propio identificador para comprobar si realmente los discursos cercanos en el dendrograma hablan de los mismos temas. Para ello, en la Figura 12 se muestra el clúster rojo mostrado en la Figura 11 junto con el identificador de cada discurso. Para ello se ha asignado un identificador a cada discurso de forma que los incluidos en el ranking A tienen un identificador entre 0 y 160, y a partir de ese número se trata de discursos de ranking B.

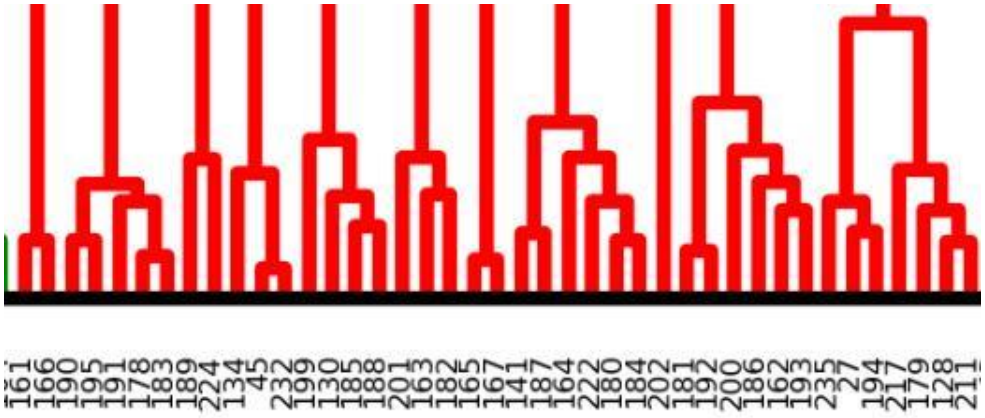

Figura 12. Dendrograma con identificadores de discurso

El topic que hace que los discursos que aparecen en la Figura 12 se diferencien del resto, es el topic 9, ya que dicho topic presenta una mayor importancia en estos discursos que en el corpus. Se puede comprobar que los que contienen este topic como el más influyente, aparecen muy cercanos en la gráfica. Por ejemplo, observando la Figura 12, los discursos más cercanos son el 45 y el 232, los cuales poseen respectivamente un 0,353 y un 0,356 de relevancia en el topic 9. Éste es un valor muy elevado para no ser un topic general (más de una tercera parte de dichos textos trata de ese topic)

En la Figura 13 se muestra una zona del fichero CSV (en el que se encuentran todos los discursos ordenados por identificador) en la que se ven diversos discursos de ranking B, y se comprueba que en general el topic 9 está presente en ellos. Aunque en este clúster predominen los discursos de ranking B, se puede apreciar que existen diversos discursos de ranking A, aunque son un porcentaje reducido del total de discursos del clúster. Uno de ellos es el 27. En la Figura 14 se comprueba que, aunque ese discurso posea ese topic, los discursos cercanos en ranking no lo poseen, por lo que se corresponde con la Figura 12 en la que se visualiza que la mayoría de discursos de este clúster son del ranking B.

201	300%20conn/announcemen	8	0.5825564172	9	0.1910773547	2	0.1645726092
202	300%20conn/senator.txt	8	0.3643566444	6	0.3072581004	9	0.283490749
203	300%20conn/spring.txt	8	0.5403911927	2	0.2387469991	4	0.156160678
204	400%20maryland/Convoca	8	0.3727176257	4	0.2510871286	2	0.1232901247
205	400%20maryland/2015%20	8	0.4301431197	6	0.2147690047	0	0.1436014677
206	400%20maryland/About%2	8	0.5687919703	0	0.1905976809	9	0.1141794991
207	400%20maryland/State%20	6	0.5315717451	8	0.3284237619	1	0.0738593577
208	300%20georgetown/2011gl	2	0.291453609	4	0.2655648165	8	0.2067711512
209	300%20georgetown/2001in	2	0.2891261543	8	0.2505580213	5	0.1676000859
210	300%20georgetown/2012co	4	0.6463256953	8	0.1956215596	2	0.1148287331
211	300%20georgetown/2003re	8	0.3832856786	9	0.168172753	2	0.156398497
212	300%20georgetown/2008ch	4	0.3797427908	2	0.3043562785	8	0.1716564988
213	300%20georgetown/2001de	8	0.3446364863	4	0.3383388881	2	0.1484909697
214	300%20georgetown/2008co	4	0.4149256478	6	0.1582851188	2	0.1512367733
215	300%20georgetown/2007se	8	0.2617238417	2	0.2274839061	0	0.1561284272
216	300%20georgetown/2012co	4	0.4516487859	2	0.2825481679	8	0.253813178
217	300%20georgetown/2010fu	8	0.3977336154	4	0.2831963164	9	0.1342908145
218	300%20georgetown/2010ob	4	0.2604873618	2	0.2307415381	8	0.2164412322
219	300%20georgetown/2009m	2	0.2954504669	4	0.2853417443	5	0.26627612
220	300%20georgetown/2012co	4	0.6280592894	2	0.2380025149	8	0.1007030787
221	400%20arkansas/Rememb	4	0.5392890612	5	0.2236756779	8	0.1019046882
222	400%20arkansas/UALR%2	8	0.5179210387	2	0.2464711743	9	0.2233071997
223	400%20arkansas/Needed:%	2	0.5626881696	8	0.2875708536	9	0.0684772197
224	400%20arkansas/Regional	9	0.3576344996	2	0.3098896603	8	0.1836431938
225	400%20arkansas/Reconcil	7	0.395147606	2	0.2116310435	8	0.1678373031
226	400%20arkansas/You%20h	8	0.3096480776	7	0.227759254	2	0.2166575839
227	500%20kent/Akron%20Rou	6	0.2594858789	2	0.2090696371	8	0.198852074
228	500%20kent/Inaugural%20	8	0.5669387657	4	0.2345161442	2	0.0899047871
229	500%20kent/%22The%20H	8	0.623357666	4	0.1486161616	2	0.1304208737
230	500%20kent/Bowman%20S	8	0.4705775213	4	0.1909435676	2	0.1063342035
231	500%20kent/May%204%20	4	0.4599166229	8	0.4002800036	2	0.0535973802
232	300%20temple/safety.txt	9	0.358216665	8	0.3263752024	3	0.1614058413
233	300%20temple/thank%20yo	8	0.361893353	4	0.2469263816	3	0.1984899857
234	300%20temple/new%20fron	8	0.4148257047	3	0.2529045844	2	0.0945375759
235	300%20temple/convocation	3	0.3061018659	8	0.2825999573	9	0.2318775246
236	300%20temple/board%20o	8	0.528929786	3	0.2121093007	9	0.0821703177

Figura 13. Topic 9 en discursos de ranking B

21	primeras/58%20uio/2012/ar	5	0.2889261667	8	0.2862998317	2	0.2562933522
22	primeras/58%20uio/2012/op	5	0.5263673277	8	0.2967258554	2	0.1615264706
23	primeras/3%20mit/Remark	1	0.2283078007	0	0.212973254	8	0.2013687443
24	primeras/3%20mit/Inaugur	8	0.4953691057	4	0.18825799	0	0.1356471813
25	primeras/3%20mit/Remark	8	0.3445258583	4	0.3432982903	2	0.1379376468
26	primeras/3%20mit/Remark	8	0.5376310699	4	0.2941923543	2	0.1042051403
27	primeras/3%20mit/Memori	4	0.3963675782	8	0.3360468808	9	0.2074950281
28	primeras/3%20mit/On%20t	4	0.5497655384	8	0.2792078181	2	0.141428024
29	primeras/3%20mit/Dedicat	4	0.5826685195	1	0.1224863911	2	0.1176373722
30	primeras/3%20mit/Remark	4	0.4709282586	8	0.2643991696	2	0.2076870026
31	primeras/3%20mit/MIT-Wh	8	0.334914781	1	0.2687615099	2	0.1776683257
32	primeras/3%20mit/Presid	4	0.6136873997	8	0.2507335618	2	0.0842004609
33	primeras/3%20mit/Remark	4	0.6036874751	8	0.1830347202	7	0.0752121405
34	primeras/3%20mit/Presid	4	0.5390067237	8	0.2230187182	2	0.0642359337
35	primeras/8%20columbia/we	8	0.4747635306	6	0.1532045098	2	0.1430651275
36	primeras/8%20columbia/20	2	0.4518039631	4	0.3233461828	8	0.0796305397

Figura 14. Topic 9 en el ranking A

Puede concluirse, por tanto, que existe un topic relevante en cada uno de los clústeres considerados, que se trataría del topic 4 para los discursos de las universidades de la zona derecha del clúster verde y del topic 9 para las universidades del clúster rojo.

Si aumentamos el nivel de detalle y nos centramos en las palabras con mayor importancia dentro de cada uno de estos topic, las palabras del topic mayoritario en discursos de ranking inferior son campus, región y trabajo. En cambio, las palabras, presentes en los discursos de las universidades de ranking superior son mundo, hoy, vida, retos, servicios y honor.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Uno de los objetivos de este trabajo era crear un repositorio textual con discursos de rectores. Esto se ha realizado satisfactoriamente con el crawler Httrack, programa que guarda los resultados de forma local, por lo que se han guardado múltiples discursos en una carpeta local del ordenador. Tras utilizar los modos de crawling semiautomático y manual, el número total de discursos obtenidos ha sido de 3105. Posteriormente, se han aplicado los mismos métodos para la obtención de discursos en el mismo idioma y presentes en un rango determinado en el ranking de Shanghái. Tras dicha ejecución se han seleccionado 276 discursos, que han sido validados y pre-procesados para ser analizados de forma óptima.

Se ha aplicado el programa Mallet a los discursos seleccionados para realizar topic modeling, que era el objetivo de la segunda etapa del trabajo. De esta forma se han obtenido los temas sobre los que tratan los discursos.

Finalmente, se han empleado algoritmos de clustering para obtener diferentes grupos, a partir de los temas tratados por cada discurso. Se ha comprobado que hay una cierta semejanza entre los discursos que ocupan un lugar cercano en el ranking. Se ha cuantificado esta semejanza empleando un perceptrón multicapa (70% de aciertos). Además, se ha visualizado un dendrograma con el método aglomerativo (clustering jerárquico). El resultado con el que mejor se visualiza dicha diferencia en un dendrograma ha sido empleando la función de enlace Ward y 10 topics. Este número fue elegido debido a que para un número menor los topics encontrados eran demasiado generales y para un número mayor éstos eran demasiado específicos para cada discurso.

Un objetivo adicional era ejecutar programas en la nube y comparar los resultados con los programas que se ejecutan en local. Para la ejecución de aplicaciones en la nube se ha utilizado Bluemix. Debido a que para realizar el trabajo era preferible disponer del repositorio en local, que dicho repositorio no tenía un tamaño excesivo, y que los algoritmos de análisis de texto eran específicos (no había un servicio que ofreciera topic modeling en la plataforma), ha prevalecido la forma de ejecución local.

La metodología seguida, tanto para obtener archivos de diferentes modos con Httrack, como para analizar los textos con Mallet y obtener y visualizar los diversos grupos resultantes con algoritmos de clustering en Python, puede aplicarse a cualquier documento de texto. Para escoger otra temática, solo habría que cambiar los filtros del crawler, y las búsquedas realizadas con anterioridad con motores de búsqueda para obtener los enlaces donde se encuentran dichos documentos. Además, habría que decidir qué stopwords excluir de los documentos a tratar. Una vez realizadas dichas modificaciones, el procedimiento funcionaría de la misma manera.

5.2 Trabajo futuro

Existen diversas formas con las que se podría continuar el trabajo. En primer lugar, utilizando las mismas herramientas empleadas durante el trabajo, se podrían buscar discursos en diversos idiomas, teniéndolo en cuenta a la hora de extraer los temas de los que tratan y buscar relaciones entre ellos. De esta forma se podrían comparar los resultados y comprobar si las diferencias obtenidas en este trabajo con discursos de distinto rango en inglés, son extrapolables a cualquier conjunto de discursos de universidades con distinto rango, independientemente del idioma en el que hayan sido pronunciados los discursos.

Además, se podría ejecutar tanto el algoritmo LDA para obtener los temas como los algoritmos de clustering en la plataforma Spark. Al usar esta plataforma, se podría aumentar considerablemente el número de documentos a analizar, por lo que, aparte de incrementar el número de discursos, se podrían analizar todos los documentos obtenidos en las páginas de múltiples universidades, y de esta forma comparar si la diferencia obtenida en el trabajo radica solamente en los discursos de los rectores.

Referencias

[1] Trupti V. Udapure, Ravindra D. Kale2, Rajesh C. Dharmik3, (2014)"Study of Web Crawler and its Different Types ", IOSR Journal of Computer Engineering. Volume 16, Issue 1, Ver VI, PP01-05.

[2] Edwards, J., McCurley, K., & Tomlin, J. (2001, April). An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the 10th international conference on World Wide Web* (pp. 106-113). ACM.

[3] «Comparison of existing open-source tools for Web crawling and indexing of free Music». http://es.scribd.com/doc/123153248/Comparison-of-existing-open-source-tools-for-Web-crawling-and-indexing-of-free-Music. [Accedido: 10-ene-2016].

[4] «HTTrack Website Copier - Offline Browser». http://www.httrack.com/html/index.html. [Accedido: 10-ene-2016].

[5] «GNU Wget». https://www.gnu.org/software/wget/. [Accedido: 10-ene-2016].

[6] «FrontPage - Nutch Wiki». http://wiki.apache.org/nutch/. [Accedido: 10-ene-2016].

[7] «IBM Bluemix - Plataforma de desarrollo de apps en la nube de próxima generación». https://console.ng.bluemix.net/. [Accedido: 10-ene-2016].

[8] «IBM Watson: What is Watson?». http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html. [Accedido: 10-ene-2016].

[9] «Software Development Kits (SDKs) | AlchemyAPI». http://www.alchemyapi.com/developers/sdks. [Accedido: 10-ene-2016].

[10] Steyvers, M., Griffiths, T.: Probabilistic topic models. In: Latent Semantic Analysis: A
Road to Meaning. Lawrence Erlbaum (2005)
http://psiexp.ss.uci.edu/research/papers/SteyversGriffithsLSABookFormatted.pdf

[11] «Latent Dirichlet Allocation - Wikipedia, the free encyclopedia» https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation [Accedido: 13-ene-2016].

[12] «Dirichlet distribution - Wikipedia, the free encyclopedia». https://en.wikipedia.org/wiki/Dirichlet_distribution. [Accedido: 13-ene-2016].

[13] «About MALLET». http://mallet.cs.umass.edu/about.php. [Accedido: 13-ene-2016].

[14] «Getting Started with Topic Modeling and MALLET | Programming Historian». http://programminghistorian.org/lessons/topic-modeling-and-mallet. [Accedido: 13-ene-2016].

[15] «Stanford Topic Modeling Toolbox». http://nlp.stanford.edu/software/tmt/tmt-0.4/. [Accedido: 13-ene-2016].

[16] L. P. Coelho y W. Richert, *Building Machine Learning Systems with Python - Second Edition*, 2 edition. Packt Publishing - ebooks Account, 2015.

[17] «2.3. Clustering — scikit-learn 0.17 documentation». http://scikit-learn.org/stable/modules/clustering.html. [Accedido: 13-ene-2016].

[18] Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

[19] «Empirical evaluation of the impact of k-means initialization — scikit-learn 0.17 documentation». http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_stability_low_dim_dense.html#example-cluster-plot-kmeans-stability-low-dim-dense-py. [Accedido: 11-ene-2016].

[20] «Hierarchical clustering (scipy.cluster.hierarchy) — SciPy v0.16.1 Reference Guide». http://docs.scipy.org/doc/scipy-0.16.0/ reference/cluster.hierarchy.html#module-scipy.cluster.hierarchy. [Accedido: 13-ene-2016].

[21] «Added Example for Plotting Hierarchical Clustering Dendrogram by MatKallada · Pull Request #3464 · scikit-learn/scikit-learn · GitHub». https://github.com/scikit-learn/scikit-learn/pull/3464. [Accedido: 13-ene-2016].

[22] «ARWU World University Rankings 2015 | Academic Ranking of World Universities 2015 | Top 500 universities | Shanghai Ranking - 2015». http://www.shanghairanking.com/ARWU2015.html. [Accedido: 15-ene-2016].

Anexos

A. Dendrogramas según la función de enlace

En este anexo se visualizarán diversos dendrogramas variando la función de enlace que se emplea. En el trabajo se ha utilizado la función Ward y se ha explicado que es en la que mejor se visualizaban los patrones en los dendrogramas. Además de dicha función, se ha ejecutado clustering jerárquico con las funciones de enlace simple, completa y promedio. Dichas funciones ya han sido explicadas durante el trabajo. Para comparar resultados, las figuras que se mostrarán han sido creadas con 10 topics, al igual que la Figura 7, que mostraba los resultados para la función Ward.

La Figura 15 muestra el dendrograma con la función de enlace completa, la Figura 16 con la función promedio y la Figura 17 con la función simple. Se visualiza como las mayores distancias entre clústeres se encuentran en la Figura 15 y las distancias se reducen tanto en la Figura 16 como en la 17. Esto es lo esperado ya que en la Figura 15 se emplea la función de enlace completa, la cual escoge como distancia entre dos clústeres la mayor distancia entre cualquier punto de un clúster con cualquiera del otro. Por lo tanto, dichas distancias serán mayores que si se cogen haciendo el promedio o escogiendo la distancia más corta.

Además de las distancias, en las figuras se visualiza como con estos métodos no se reconocen patrones de la forma en la que se ha realizado con la función Ward.

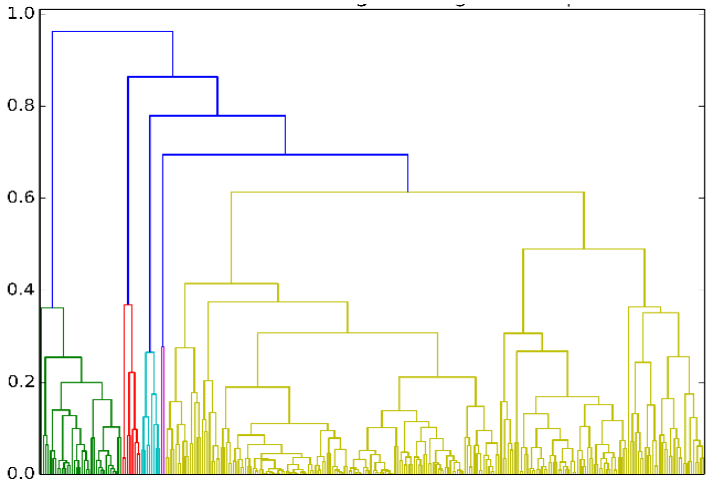

Figura 15. Función de enlace completa

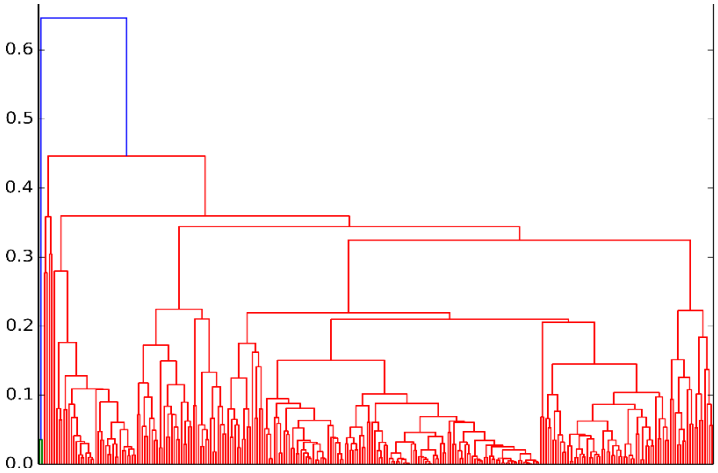

Figura 16. Función de enlace promedio

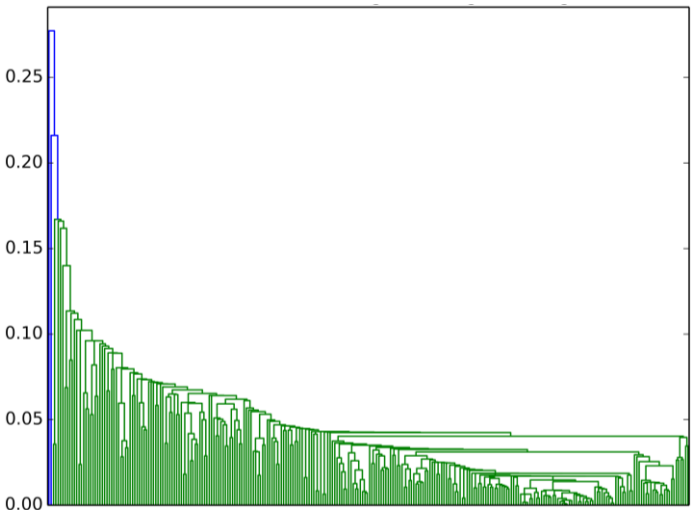

Figura 17. Función de enlace simple

B. Dendrogramas variando parámetros de Mallet

Durante el trabajo se han explicado diversos parámetros del programa Mallet. Finalmente se han empleado 10 topics y 1000 iteraciones para el muestreo de Gibbs. Para realizar dicha elección, se han realizado múltiples ejecuciones y ésos han sido los valores que mejor resultado han obtenido.

En la Figura 7 se observó el dendrograma con la función Ward y empleando 10 topics. En la Figura 18 se muestra el dendrograma, empleando también la función Ward, utilizando 4 topics. En la Figura 19 se emplean 100 topics.

En la Figura 18 se observa como la distancia entre clústeres es considerablemente mayor que al utilizar 10 topics. Esto es debido a que, con tan solo 4 topics, el porcentaje que contenga un discurso de su principal topic será un porcentaje elevado (ya que el máximo es 1, y solamente hay que repartir porcentajes entre 4 topics). Por lo tanto, la diferencia entre ese discurso y otro cuyo topic principal no sea el mismo, será mayor que en la ejecución con 10 topic.

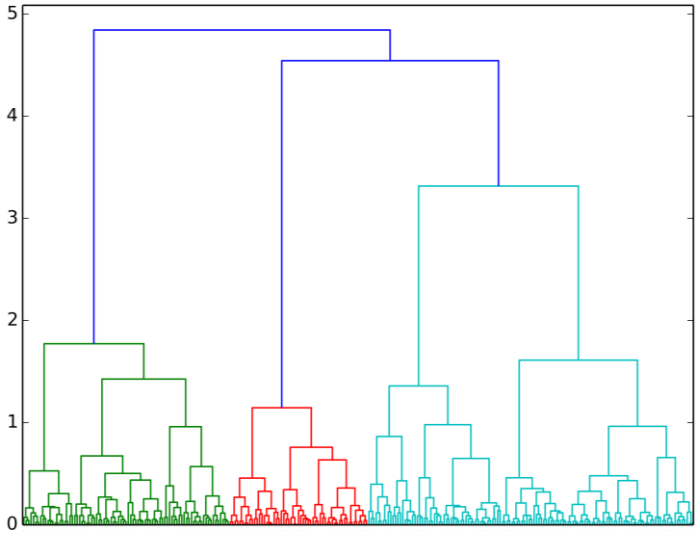

Figura 18. Dendrograma empleando 4 topics

En la Figura 19 se observa que al emplear 100 topics la distancia entre los clústeres es considerablemente menor que al emplear 4 ó 10. Esto es debido a que con 100 topics, cada discurso dividirá el valor máximo de importancia (el cual es 1) entre las 100 dimensiones, según la importancia que tenga cada topic en él. Al tener valores pequeños en cada dimensión, los discursos con topics parecidos, estarán más cerca. Además, en la figura también se visualiza que no se encuentran de forma adecuada patrones, ya que la mayoría de los discursos se encuentran en un mismo sub-clúster. Esto es debido a que, al emplear demasiados topics, estos serán muy individuales, por lo que estarán en pequeñas proporciones en los discursos. Sin embargo, los topics generales seguirán teniendo

importancia en múltiples discursos, por lo que se les dará más importancia a dichos topics que a los específicos.

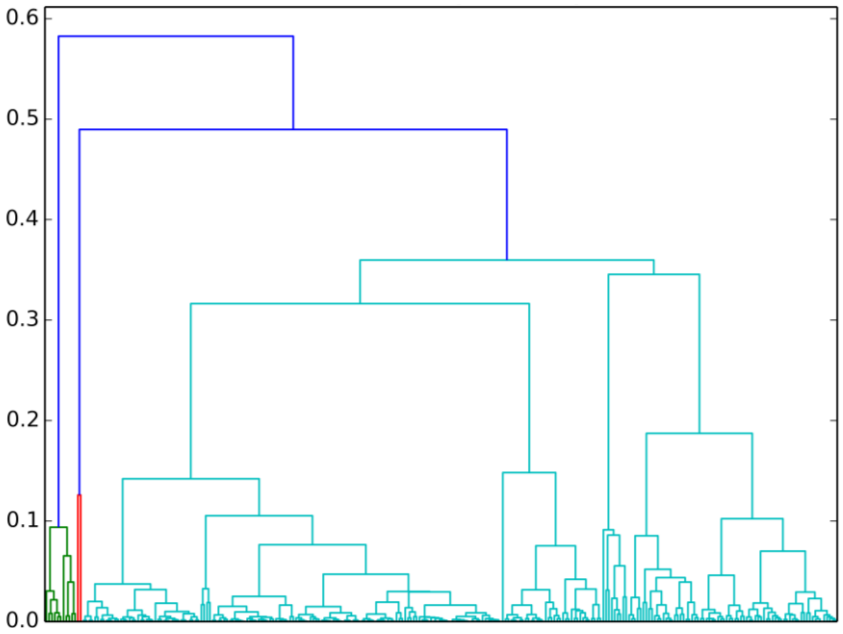

Figura 19. Dendrograma con 100 topics

En la Tabla 2 se ha mostrado el tiempo que tarda el programa Mallet según el número de iteraciones elegidas en el muestreo de Gibbs. Posteriormente, se ha indicado que se ha elegido el número 1000 debido a que es el valor óptimo entre tiemplo empleado y calidad de los topics que encuentra. Se han realizado múltiples ejecuciones para probar el resultado variando dicho número. En las Figuras 20 y 21 se muestra la importancia que posee cada topic, tras 100 y 10 000, respectivamente, de muestreo de Gibbs.

0	4.4991
1	4.40645
2	3.37097
3	5.26782
4	4.21018
5	4.39542
6	3.75623
7	3.50477
8	5.2199
9	5.29493

Figura 20. 100 iteraciones

0	1.52702
1	0.25205
2	0.16566
3	0.58702
4	0.11398
5	0.7709
6	0.11651
7	0.23369
8	0.41987
9	0.73651

Figura 21. 10 000 iteraciones

En la Figura 20 se puede apreciar que 100 iteraciones no son suficientes para que el algoritmo halle una importancia que pueda parecerse al contenido real de los textos. En la Figura 21 se visualiza que sí se ha hallado la importancia de cada topic, pero es posible,

que ese valor se haya sobreajustado, ya que los topics con muy poca importancia propiciarán que la mayoría de discursos posea como topics con mayor importancia a los topics generales, y que, por tanto, la mayoría de los discursos se incluyan en un mismo sub-clúster (al igual que ocurría al emplear demasiados topics). Para contemplar mejor el resultado con dichas iteraciones, en las Figuras 22 y 23 se muestran los respectivos dendrogramas. La Figura 22 es similar a la Figura 18, ya que el hecho de emplear pocas iteraciones de Gibbs es similar a emplear pocos topics, debido a que, en ambos casos, cada topic tendrá una importancia similar en el corpus (la cual será alta).

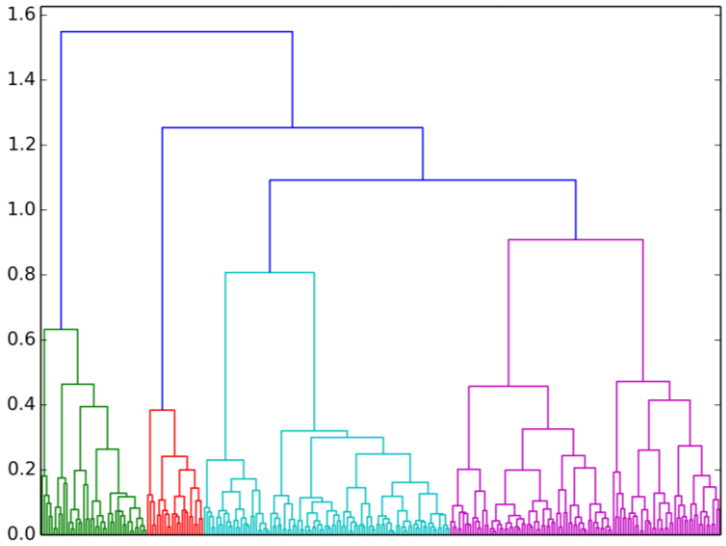

Figura 22. Dendrograma con 100 iteraciones para el muestreo de Gibbs

Del mismo modo, la Figura 23 es similar a la Figura 19, ya que el emplear demasiadas iteraciones de Gibbs se asemeja a utilizar demasiados topics, debido a que, en ambos casos, existirán topics con porcentajes en el corpues escasos, que propiciarán que la mayoría de discursos estén en el mismo clúster, formado por los topic generales al corpus.

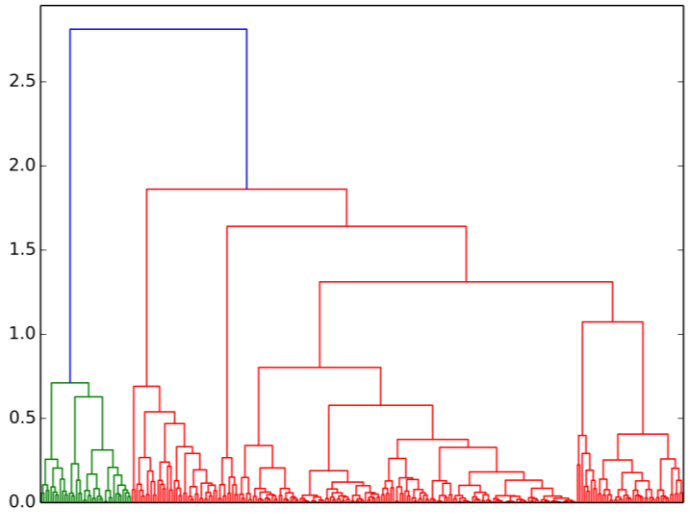

Figura 23. Dendrograma con 10 000 iteraciones en el muestreo de Gibbs