Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

Título

Análisis cronológico de opinión en diarios utilizando extracción de tópicos y word-embeddings.

por
Maximiliano Ezequiel Tejerina

Trabajo Especial de la carrera Licenciatura en Ciencias de la Computación

Directores: Martin Dominguez, Andrés Matta.
Tribunal: Laura Alonso Alemany, Milagros Teruel.

Resumen

Este trabajo tiene como objetivo el análisis del discurso de dos diarios argentinos La Nación y Página 12, intentando capturar el comportamiento de dos editoriales con opiniones políticas antagónicas. Para lograr el análisis se recopilaron notas periodísticas de ambas fuentes, que traten el tema de la “Reforma Laboral”, desde el año 1995 al 2021.
Con este conjunto de artículos, se separó en períodos presidenciales, y se utilizaron herramientas de extracción de temas, para analizar la manera en que cada editorial trata el tema elegido. Adicionalmente, se utilizaron técnicas de “word embeddings” para analizar la distancia entre ambos discursos en los diferentes períodos temporales..

Para llevar a cabo la tarea, se evaluaron tres modelos de aprendizaje automático con múltiples configuraciones, con el objetivo de encontrar así el mejor rendimiento.

En este sentido, en una primera instancia, con técnicas de Procesamiento de Lenguaje Natural aplicadas sobre el contenido de los artículos, se capturó la tematización y limpieza del texto.

En una segunda instancia se logró inferir un modelo LDA (Latent Dirichlet Allocation) en dos librerías de python,
Luego, construimos los “word embeddings” utilizando Fasttext para poder obtener las distancias entre las opiniones de ambos diarios.

Del análisis de los resultados obtenidos por LDA para la detección de tópicos se concluyó que, debido a la tipología de los artículos, los resultados no eran satisfactorios. Para solucionar este problema, se exploró otra técnica, Top2Vec. Los resultados obtenidos con esta nueva técnica, fueron satisfactorios, y permitieron identificar algunas particularidades en el tratamiento de las temáticas en cada periódico, las que pueden ser asociadas a agendas y marcos interpretativos diferentes.

Agradecimientos:

Primeramente agradecer a Dios por estos años de vida. A mi hermosa novia por el apoyo, estar presente, alentarme y darme ánimos en cada etapa. A mis pastores Walter, Analía y José Luis por promover que estudiar es importante. Y a mi familia. Como así, a Globant por abrirme las puertas hacia la industria y brindarme accesos para poder terminar dicho trabajo.
Muchas gracias a la facultad, por abrirme al conocimiento necesario para enfrentarme a la vida laboral. Estudiar en la FaMAF fue desafiante, con gran alegría festejaba cada paso, cada materia aprobada.
A todo el equipo de trabajo de la facultad, desde el área de biblioteca a al área de electrónica quienes ayudaron a que siempre tenga el acceso de una computadora propia, como así al área de becas que sin ese impulso económico hubiera sido imposible seguir estudiando, les estoy gratamente agradecido.
Una especial gratitud a mi profesor de secundaria Ricardo Palumbo quien me cultivó en 6to año las matemáticas. Me encaminó hacia un rumbo lleno de dicha y sabiduría, su corazón amable me ayudó a enfrentarme en los inicios académicos y vida universitaria . Y por supuesto agradecer a mis directores Andres Matta , Martín Domínguez , como así también a Juan Porta quién fue mi ayudante en estos últimos meses para poder concluir esta grandiosa etapa.
Por último, agradecer a dos factores claves, a los profesores de la diplomatura que sin el enriquecimiento de conceptos en la misma no hubiera realizado el presente trabajo y a los organizadores de CCAD por darme acceso a ATOM.

Dedicatoria:

Dedico este trabajo a todos mis seres queridos, a quienes me apoyaron a seguir estudiando por más difícil que sea la circunstancia. Cada dificultad, la pude superar por el aliento de otros. A los excelentes profesores que me han tocado a lo largo de mi carrera.
A mi abuelo, un gran hombre, que desde niño me educó a hacer lo correcto y a mi abuela quién puso su vida y tiempo para estar a mi lado.
Este trabajo está hecho en colaboración con un gran investigador y docente Andres Matta, a quien dedicó gran parte de este esfuerzo.
Dedico este trabajo a todos aquellos que creen en el esfuerzo y trabajo duro, como así a los distintos profesores que me han tocado a lo largo de estos años.

Índice

1. Introducción.

1.1 Motivación:

Dada la enorme cantidad de enlaces provistos por un diario a lo largo de los años y a la información relevante a través de dichos artículos en internet. Es muy dificultoso buscar artículo por artículo dado un tema particular, sin contar la enorme cantidad de enlaces repetidos o enlaces basura. Con enlace basura nos referimos a enlaces rotos o enlaces que dentro de la misma no contienen la estructura de cuerpo periodístico. En el siguiente trabajo se estudió dos diarios por lo que se recopiló y analizó cada url de estos.

Entonces nos motiva tener una base de datos o data frame donde se encuentren todos los enlaces de una determinada búsqueda ya sea en el buscador de los diarios, api o en el buscador google. Este conjunto dado por la recopilación de enlaces a artículos tiene como fin la extracción de texto, el guardado y un estudio sobre el texto desarrollado con el aprendizaje automático.

1.2 Machine learning:

Machine Learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto consiste en dejar que los algoritmos descubran patrones en conjuntos de datos.

Al almacenar digitalmente datos (ya sea imagen, texto o números) pueden servir como input para los modelos de Machine Learning. Al detectar patrones en esos datos, los algoritmos aprenden y mejoran su rendimiento en la ejecución de una tarea específica.

En general se subdivide en dos ramas importantes:

- El aprendizaje supervisado supone que partimos de un conjunto de datos etiquetado previamente, es decir, conocemos el valor del target para el conjunto de datos que disponemos.
- Por el contrario, el aprendizaje no supervisado, parte de un conjunto de datos del que no se tiene un conocimiento a priori, siendo el objetivo en este tipo de análisis la comprensión de los datos o la transformación automática de éstos.

Un ejemplo de aprendizaje supervisado sería la construcción de un modelo de reingresos en hospitalización partiendo de un conjunto de datos previo de los que conocemos si el paciente reingresó o no (el atributo que nos indique la condición de reingreso en el conjunto de datos original sería la etiqueta).

Como ejemplo de aprendizaje no supervisado: Una agrupación de los datos según su similaridad (clustering). En genética, por ejemplo, agrupar patrones de ADN para analizar la biología evolutiva como así también en sistemas de recomendación, que consisten en agrupar usuarios con patrones de visualización similares para recomendar contenidos similares. O en detección de anomalías, incluida la detección de fraudes o la detección de piezas mecánicas defectuosas.[2]

Dentro de la rama del aprendizaje automático, se encuentra el procesamiento del lenguaje natural o NLP, dentro de esta área se encuentra nuestro problema que abordaremos (LDA).

En el procesamiento del lenguaje natural, la asignación latente de Dirichlet (LDA) es un modelo probabilístico generativo, el cual permite explicar conjuntos de observaciones mediante grupos no observados que explican por qué algunas partes de los datos son similares. Más adelante detallaremos, como así también describiremos varios ejemplos de LDA. (Capítulo 5)

1.3. ¿LDA qué es? ¿y para qué lo usaremos?

Comenzaremos con una breve introducción de nuestra aplicación de LDA para poder adentrarnos al problema en cuestión. En el aprendizaje automático, una aplicación de LDA es descubrir temas en una colección de documentos y luego clasificar automáticamente cualquier documento individual dentro de la colección en términos de cuán "relevante" es para cada uno de los temas descubiertos. Se considera que un tema es un conjunto de términos (es decir, palabras o frases individuales) que, tomados en conjunto, sugieren un tema compartido.

Nuestro trabajo se desarrolla en el ámbito de economía, por lo tanto, se obtiene un conjunto de temas relacionados al área. Más adelante estudiaremos a LDA y algunas herramientas para obtener tópicos, para ello se estudiaron dos librerías, Gensim [20] y Sklearn [21]. Para luego adentrarnos en top2vec.

1.5. Problema de scraping y solución:

Al tener que buscar en un buscador dentro de dos diarios lanacion* y página 12 ** un tema específico, en nuestro caso artículos relacionados a la reforma laboral, se obtuvieron miles de artículos con ciertos parámetros de búsqueda como REFORMA+LABORAL, “+REFORMA+LABORAL”, “REFORMA LABORAL”. En tales búsquedas se obtuvieron conjuntos disjuntos con muy poca o mucha intersección. Para poder profundizar en el tiempo se decidió en el caso de la nación, cubrir los primeros 5 mil enlaces. Más tarde se detallará cómo fueron las búsquedas (capítulo 2.3).

Mientras que en página 12, se usó una API específica provista por la misma ***. Debido a este motor de búsqueda nos proporcionó un buen resultado buscando la palabra exacta.

Luego, para poder obtener los enlaces sin repeticiones usamos el siguiente pseudocódigo (i), pero primeramente se observó que nuestro algoritmo recolectó publicidades por lo que se decidió eliminarlas de la lista: (ii)

(ii) para cada artículo en lista de direcciones url do
si no hay ‘index’ o ‘resultado’ o ‘search ‘ en articulo entonces
guardar en lista nueva este artículo.

(i) lista_articulos_sin_repeticiones = list(set(lista de direcciones url)

En este trabajo se recopiló 2500 enlaces de página 12 y 3659 enlaces del diario La Nación sobre la búsqueda: “reforma laboral” desde los años 1995 al 2022.
Previamente se clasificó por periodos presidenciales, para poder analizar dichas urls posteriormente.
Se usaron herramientas de extracción de artículos de diarios propuestas por SELENIUM (nos dedicaremos a esta brillante herramienta en breve) desde la sección de economía de cada diario para posteriormente ser analizado por un experto de dominio.

Se utilizaron librerías de python para el extracción de texto:
"Newspaper es una increíble biblioteca de Python para extraer y curar artículos"
tuiteado por Kenneth Reitz.[3]

Previamente a la extracción del contenido de cada links , se obtuvieron las URLs de cada diario en un periodo de tiempo especificado. Para ello, se tuvo que indagar por cada búsqueda la cantidad de artículos a elegir de cada diario, es decir, elegir el

número óptimo de enlaces para cubrir todas las etapas presidenciales. Luego de observar que eran pocos los artículos para página 12 en los años anteriores a 1999 se recurrió a una búsqueda por google.
La importancia de este trabajo se enfoca a un análisis morfológico dentro de la rama de la economía argentina en el lapso de tiempo entre 1995 y 2022 para los períodos presidenciales existentes dentro de este tiempo.

1.6. Dataset

Al trabajar con enlaces primeramente, se recopiló en formato .csv listas de URLs las cuales provenían de tres búsquedas distintas en cada diario. Más adelante se mencionará sobre dichas búsquedas.

Por otro lado, se trabajó sobre las listas de urls con un software de scraping para extracción de artículos, en el siguiente capítulo abordaremos el tema. Gracias a esto se armó un dataset robusto. Este dataset será utilizado para predecir tópicos al final de este trabajo, a fin de obtener un modelo de fasttext y obtener una curva relacionada entre ambos diarios por cada periodo (capítulo 9).

*https://www.lanacion.com.ar/ página oficial del diario La Nación
**https://www.pagina12.com.ar/ página oficial del diario pagina12

***(https://www.pagina12.com.ar/buscador/index.php API búscador

2. Extracción y scrapping.

2.1. Extracción de la información

La Extracción de Información (EI o generalmente referida como IE, por Information Extraction) es una tarea del Procesamiento de Lenguaje Natural (PLN) que consiste en la extracción automática de información de un medio no estructurado obteniendo información estructurada como entidades, relaciones entre entidades y atributos que describen entidades. [4]

Esto es importante ya que tener la información estructurada nos permite operar con ella para realizar consultas, organizarla y analizarla.

En el trabajo aquí presentado se utilizó la extracción de artículos periodísticos de dos diarios argentinos. Tales diarios tienen una forma diferente de exponer la información.

Por un lado el diario La Nación, donde se logró alrededor de 3659 links. Por el otro, la extracción de artículos periodísticos del diario de página 12 con 2560 links.

Cabe aclarar, (como lo dicho en el capítulo 1.5) que para conseguir la mayor cantidad de links en el periodo de 1995 a 2002 y debido a que estas fechas no se apreciaban en la búsqueda de las páginas oficiales para la búsqueda en página 12, se realizaron búsquedas en google, una para el año 1995 , otra para el año 1996, para el año 1997 , escalando a través de los años hasta 2002 , filtrando enlaces basura y guardando los enlaces en la fecha en cuestión.

Ambos diarios fueron clave debido a su morfología en representación semántica a la hora de analizar una misma noticia en dos diferentes fuentes.

2.2 Extracción de Información sobre URL

En este apartado se aclara sobre las herramientas que se utilizaron para la extracción de URL's. Primeramente se logró ubicar los links mediante SELENIUM. [5]

En el núcleo de Selenium se encuentra Selenium WebDriver, una interfaz para escribir instrucciones que funcionan indistintamente en todos los navegadores.
Es el sucesor de Selenium RC. Selenium WebDriver acepta comandos y los envía a un navegador. Esta herramienta corre a través de un controlador de navegador específico, (particularmente se utilizó un controlador para Chrome), que envía

comandos al navegador y recupera los resultados. En nuestro caso, los comandos más utilizados fueron búsqueda de ciertos atributos dentro de un html como etiquetas , clases , y hasta atributos javascript. [5]

La mayoría de los controladores de navegador inician y acceden a una aplicación de navegador (como Firefox, Google Chrome, Internet Explorer, Safari o Microsoft Edge).

Luego de ubicar la “búsqueda” en el buscador del diario la nación, se accedió a buscar 3 frases en la sección de economía; “Reforma Laboral”, Reforma&Laboral, +Reforma+Laboral .

- “Reforma Laboral”: Denota la búsqueda exactamente en secuencia y en ese orden.
- Reforma&Laboral: Denota la búsqueda que no importa si están en secuencias, si no que ambas palabras pertenecen al mismo artículo.
- +Reforma+Laboral: Denota la búsqueda de que al menos una de las palabras esté contenido en el artículo.

Por otra parte, en la API de página 12 se realizó la búsqueda de otra manera utilizando el motor selenium, esto se debe a que la misma api proveía una muy excelente búsqueda.

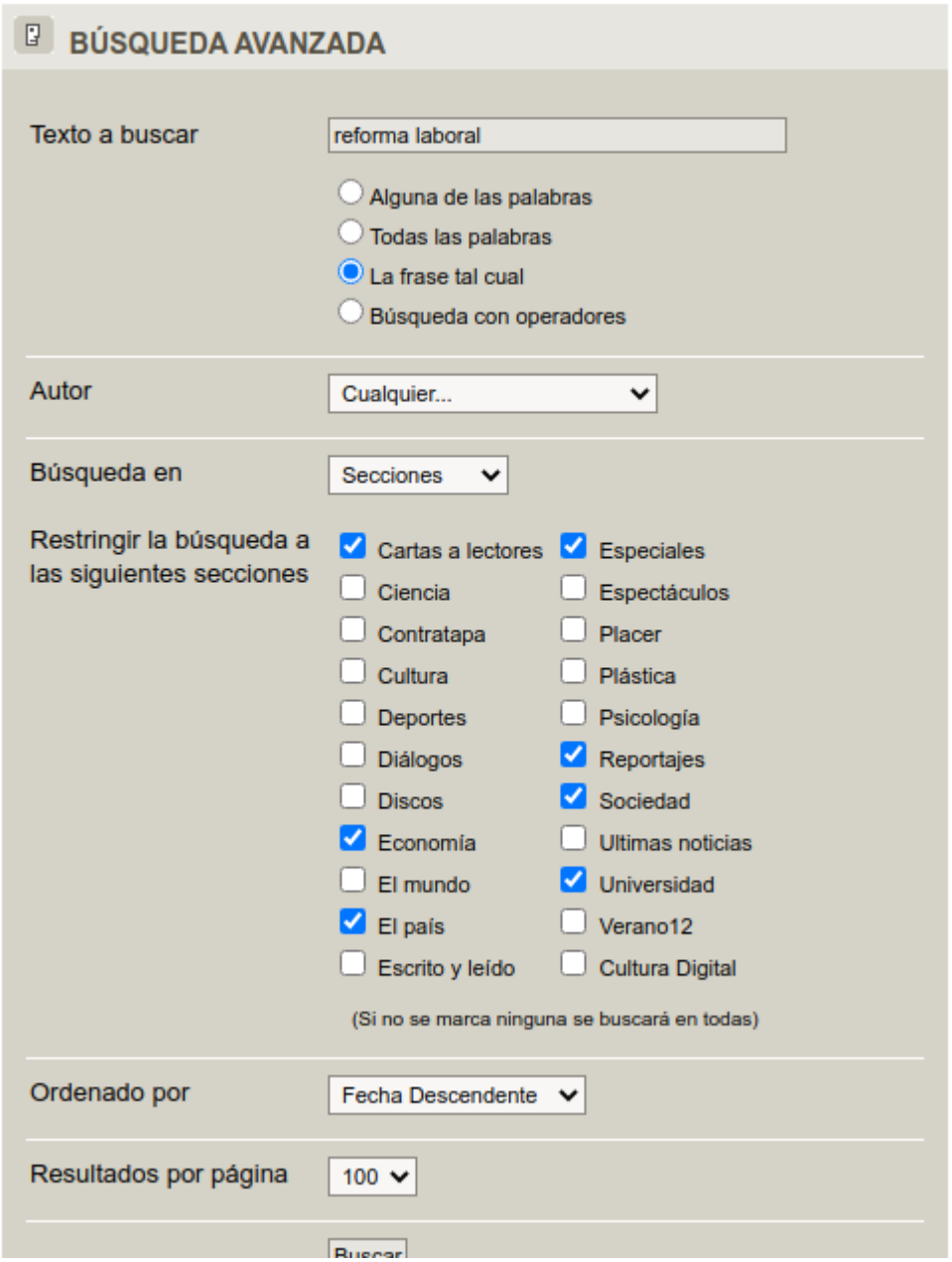


imagen 2.2.1

En la imagen 2.1.1 notamos la búsqueda realizada. Notar que los temas a interés están dentro de los sectores de: “Carta a lectores”, ”Economía”, ”El país”, ”Especiales”, ”Reportajes”, “Sociedad”, ”Universidad”. Estas pautas fueron expuestas por el experto de dominio. Así mismo se realizaron las 3 búsquedas anteriormente mencionadas.

2.3 Extracción de Información: artículos.

En este apartado se mencionará la importancia de la librería: “NewsPaper 3k”, inspirado en las solicitudes por su simplicidad y potenciado por lxml por su velocidad.
A continuación marcaremos características importantes de la misma:

- Marco de descarga de artículos de subprocesos múltiples.
- Identificación de URL de noticias.
- Extracción de texto de html.
- Extracción de imagen superior de html.
- Toda la extracción de imágenes de html.
- Extracción de palabras clave del texto.
- Extracción de resumen del texto.
- Extracción de autor del texto.
- Extracción de términos de tendencias de Google.
- Funciona en más de 10 idiomas (inglés, chino, alemán, español, árabe, …).

la reciente tabla se puede observar en [6]

Además de usar newspaper para el scraping de los artículos. Se optó por representar los datos en tablas con la librería de pandas.
Estas tablas creadas contienen los siguientes atributos:

Title	Atributo que contiene los títulos de los artículos.
Text	Contiene el cuerpo del artículo.
Fecha	Este atributo indica la fecha de publicación.
Url	Contiene el enlace o URL (Uniform Resource Locator) de dicho artículo.
Búsqueda	En qué medio se buscó (google, pagina12 o la nación).
Sirve (s)/ No sirve (n)	Este atributo especifica los niveles donde el experto de dominio clasificó.

2.4 Errores en la extracción

Al procesar más de 500 enlaces con la librería de selenium web efectuó un error 429 de TTP.

Error mostrado en pantalla.

“ArticleException: Article `download()` failed with HTTPSConnectionPool(host='www.lanacion.com.ar', port=443): Read timed out. (read timeout=10) on URL”

El error 429 de TTP es un código de estado de respuesta HTTP que indica que la aplicación cliente ha superado su límite de velocidad o la cantidad de solicitudes que puede enviar en un período de tiempo determinado.

Básicamente al extraer la data, en un momento dado, provocó un agotamiento del tiempo de lectura. Las conexiones de las páginas de diarios ocasionalmente caducan, porque utiliza las solicitudes del módulo Python.

Estos tiempos de espera generalmente están vinculados a la fuente que está consultando. journal 3k admite un parámetro de tiempo de espera en Config(), lo que podría ayudar a prevenir futuros problemas de "tiempo de espera de lectura agotado".

Inicialización de variables

```
config = Config()
config.browser_user_agent = user_agent
config.request_timeout = 10
```

No obstante con esas líneas seguíamos teniendo el mismo problema, inclusive extendiendo el tiempo a 30 minutos.

Para solucionar esto, se decidió dividir la cantidad de búsqueda en n listas de 500 enlaces cada una. De esta manera iterando cada 15 minutos entre dichas listas y guardando este resultado en n dataframe distintos de longitud 500.

Al obtener estos n dataframe se generó un join para unir los más de 3 mil enlaces con sus respectivos atributos. Este procedimiento se usó en todas las búsquedas a excepción para la búsqueda en página 12 de google.

Por su parte en la búsqueda de google no se obtuvo mucha dificultad para obtener las url en la tabla creada con pandas.

Otro de los errores encontrados, fue en la recolección de fechas. En la minoría de links al hacer el scrapeo de la información se notó que el atributo data o fecha no fue recolectado satisfactoriamente. Lo cual ocasionó una pérdida de reubicación al separar por periodos. Se hizo una búsqueda manual de fechas. Se desestimó una parte pequeñísima del conjunto sin fecha debido a no clasificar los estándares deseados.

3. Pre procesamiento de artículos.

Nuestro objetivo es pasar de un texto (artículo) a una cadena única larga, terminando con una lista (o varias listas) de tokens limpios que serían útiles para tareas adicionales que evaluaremos en capítulos futuros.

Usar el preprocesamiento de texto es básicamente limpiar, simplificar texto y obtener una lista de palabras. A continuación nombraremos algunas de estas operaciones de procesamiento de texto habituales:

- En una primera etapa la eliminación de palabras irrelevantes o stop words.
- Uso de expresiones regulares para buscar y reemplazar cadenas de destino específicas, en nuestro caso se eliminó acentos.
- Lematización: esta operación convierte varias palabras relacionadas en una única forma canónica.
- Stemming: encontrar la palabra raíz, es de importancia para el análisis.
- Normalización de mayúsculas.
- Eliminación de ciertas clases de caracteres, como números, caracteres especiales.
- Identificación y eliminación de correos electrónicos y direcciones URL.

Todo este arsenal de conceptos se puede desarrollar con la librería ntlk de python y spacy. ¿pero, para qué nos sirve el preprocesamiento?.[7]

Un documento, de cualquier índole, está hecho no sólo de palabras, sino de un sinnúmero de relaciones semánticas que solo pueden ser decodificadas por quienes dominan ciertos códigos, cierta lingüística, etnia o cultura.

Extraer automáticamente información relevante de los textos conlleva a saber qué palabras tomar respetando la idea central de dicho texto.

Solamente algunas palabras nos interesan para realizar una tarea, osea no nos interesa conocer todos los significados de un texto. Aunque las computadoras (aún) no entienden el lenguaje natural, son excelentes leyendo superficialmente grandes cantidades de texto en segundos.

Una buena técnica para obtener la información relevante de un texto consiste en eliminar los elementos que puedan ser menos importantes (palabras vacías), y resaltar más lo que los textos tienen en común que sus diferencias.

Para lograr esta técnica se utilizaron varios conceptos y procedimientos los cuales nombraremos a continuación.

3.1 Tokenización.

La tokenización es el proceso de tokenizar o dividir una cadena de texto en una lista de tokens (o palabras). Se puede pensar en el símbolo como partes: una palabra es un símbolo en una oración y una oración es un símbolo en un párrafo.

En este procedimiento se eliminan las palabras que tienen poco interés para nosotros. El primer paso es delimitar las palabras del texto, y convertir esas palabras en elementos de una lista. Este procedimiento es conocido como *tokenización*. Implementamos esto en Python, utilizaremos una librería de procesamiento de Lenguaje Natural llamada spacy.

Una vez hecha la tokenización , sacamos previamente las mayúsculas y guardamos en una columna del dataframe. Luego se hace lo mismo sacando stopwords que se explicará la utilidad de la misma a continuación. [7]

3.2 Limpieza del texto

3.2.1 Palabras vacías

Una parte crucial del procesamiento del lenguaje natural es la limpieza de texto. Empezamos con las stop words ¿qué son? ¿cómo sacarlas?

Las stop words son palabras que no tienen un significado por sí solas, sino que modifican o acompañan a otras, este grupo suele estar conformado por artículos, pronombres, preposiciones, adverbios e incluso algunos verbos.

Las palabras vacías, en el procesamiento de datos en lenguaje natural es necesario filtrarlas antes del proceso en sí, no considerándolos por su nulo significado, si no porque son consideradas palabras sin valor argumentativo al significado.

En este trabajo se utilizó la librería spacy para reconocer y eliminar dichas palabras. Esta librería nos construye la lista de tokens pero sin incluir palabras muy comunes y poco informativas desde el punto de vista léxico, tales como conjunciones (y, o, ni, que), preposiciones (*a, en, para, por,* entre otras) y verbos muy comunes (*ser, ir,* y otros más). Se utilizó esta lista mencionada como así una lista personal que se agregó a la lista de palabras vacías.[7]

3.2.2 Puntuación :

Otra regla importante dentro del procesamiento de texto, es los signos de puntuación ya sean: “”¡?¿., etcétera.

Esto se debe a que nos interesa las palabras en sí y no los signos de puntuación.
Para ello se utilizo la libreria string el cual ya lo provee en la función “string.puntuaction”. La cual te saca los signos de puntuación. [7]

3.3 Normalizar un texto con Python.

La normalización del texto es el método de transformar el texto en una forma canónica.

Normalizar el texto antes de almacenarlo o procesarlo permite separar las preocupaciones, ya que es seguro que la entrada será coherente antes de que se realicen operaciones al respecto. La normalización del texto requiere ser consciente de qué tipo de texto se normalizará y cómo se procesa posteriormente; no existe un procedimiento de normalización de uso general.

El siguiente paso en nuestro trabajo consiste en *normalizar* el texto. Nuestro tokenizador reconoce formas como *Economía*, *economía* y ECONOMÍA como palabras distintas. Además, el documento puede tener números y palabras compuestas por caracteres alfanuméricos. Como no nos interesan estas palabras, y queremos que en nuestra lista aparezcan solamente las formas convencionales (por ejemplo, *economía*, sólo en minúsculas) debemos normalizar nuestro texto. Esto lo hacemos para cada palabra, pasándole la función lower(), la cual efectúa la transformación en minúscula.[8]

3.4 Lematización.

¿Qué es la lematización?. La lematización es un mecanismo lingüístico que lleva a las palabras de una forma flexionada (es decir, en plural, en femenino, conjugada, etc), al lema correspondiente. El lema es la forma que representa una palabra de todas las formas flexionadas de esa misma palabra. Por ejemplo, economía es el lema de económico, pero también de económicos o economías; hombre es el lema de hombres, ahorrar es lema de ahorro, ahorros.[8]

Lematizar implica estandarizar, desambiguar, segmentar.
La lematización automática se realiza a través de programas de análisis morfológico.

Hay diversos grados de lematización posible:

- una lematización puramente morfológica,
- una lematización sintáctica que tenga en cuenta el contexto en el que aparece la palabra.

Por ejemplo, en un análisis morfológico la palabra cura tendría dos lemas: el sustantivo cura de una iglesia y el verbo curar. O en economía el sustantivo peso constrata con el adjetivo peso.

En el otro extremo, en contexto sintáctico (es decir, en una oración), podemos desambiguar y optar por un único lema. Por ejemplo: “El que gobierna decretó una ley”, gobierna es sustantivo, mientras que en “la presidenta gobierna ”, gobierna es del verbo gobernar.

Para poder hacer este tipo de lematización es necesario, por lo tanto, hacer un análisis sintáctico.[9]

La lematización es una tarea propia de la Lingüística Computacional, y es útil en la tecnología aplicada a buscadores, traductores automáticos, y es muy importante para el “Procesamiento del Lenguaje Natural”.

En nuestro trabajo, a pesar de que ya tenemos una lista reducida de palabras (sin stopwords), podemos achicar aún más usando lematización. Por ejemplo, sabemos que *cambia, cambio, cambian, cambiamos* son distintas formas (conjugaciones) de un mismo verbo (*cambiar*). Y qué *terminado, terminar, terminan* y otras más, son distintas formas del vocablo *terminar*. Por lo que las diferencias entre cada variante se puede juntar y englobar una nueva variante en un mismo término. Y eso es lo que hace la lematización: relaciona una palabra flexionada o derivada con su *forma canónica* o *lema*. Y un lema no es otra cosa que la forma que tienen las palabras cuando las buscas en el diccionario.

Al usar la librería de spacy de lematización se notó que si bien en su gran mayoría del conjunto de palabras encontraba su lema apropiado, había una minoría pero palabras cruciales como “económico” que aún no encontraba su lema ”economía”.

También se investigó y encontraron varios lematizadores más como el que proporcionaba “STANZA”, este entraba en conflictos con paquetes de SpaCy. Por lo que se desinstalo e instalo STANZA. Posteriormente, como daban los mismos resultados, se decidió usar el mismo lematizador de SpaCy.

La lematización es un proceso clave en muchas tareas prácticas de PLN, pero tiene dos costos. Primero, es un proceso que consume recursos (sobre todo tiempo). Segundo, suele ser probabilística, así que en algunos casos obtendremos resultados inesperados. Este segundo es la razón por lo que tenemos palabras como “cristina” y su lematización “cristian”.

Una solución a ello es buscar stemming a cada palabra relevante de esto tratará el siguiente apartado.

3.5 Stem.

Se llama *stemming* al procedimiento de convertir palabras en raíces.
Estas raíces representan la parte invariable de palabras relacionadas. Tiene una gran semejanza con la lematización, pero los resultados (las raíces) no tienen por qué ser palabras de un idioma. Por ejemplo, el algoritmo de *stemming* puede decidir que la raíz de *amamos* no es *am-* (la raíz que) sino *amam-* (cosa que desconcertaría). Aquí va un ejemplo de *stemming*: [8]
Del texto :

“””EL PAíS El hoyo negro de los fondos reservados del menemismo ”””

Nos dará de output lo siguiente:

país’, ‘hoy’, ‘negr’, ‘fond’, ’reserv’, ‘menem’.

Para encontrar las raíces en español hemos usado la librería de Python llamada nltk. Es otra librería fundamental para el procesamiento de lenguaje natural. En nltk hay muchas funciones para tareas de este tipo en varios idiomas. En el ejemplo hemos utilizado el *Snowball Stemmer* porque funciona no sólo en inglés, sino en otras lenguas.

Además de *snowball*, nltk permite usar otros algoritmos, como Porter o Lancaster.

El *stemming* es mucho más rápido con respecto al procesamiento de la lematización. También tiene la ventaja que reconoce relaciones entre palabras de distinta clase.

Una desventaja del stemming es que sus algoritmos son más simples que los de lematización. Pueden “recortar” demasiado la raíz y encontrar relaciones entre palabras que realmente no existen (overstemming). También puede suceder que deje raíces demasiado extensas o específicas, y que tengamos más bien un déficit de raíces (understemming), en cuyo caso palabras que deberían convertirse en una

misma raíz no lo hacen. No hay mucho que hacer con eso, pero el stemming es una muy buena solución de compromiso en la mayoría de los casos.[8]

En la siguiente sección presentamos resultados de procesamiento sobre el texto.

4. Bigramas y collocations.

Las colocaciones son dos o más palabras que suelen aparecer juntas con frecuencia, como “Reforma Laboral” o “Cristina Fernandez”. Básicamente es una frase que consta de dos o más palabras, pero estas palabras coexisten más comúnmente en un contexto dado que sus palabras individuales. Como ejemplo tomamos la collocation: Reforma_laboral. Por supuesto, hay muchas otras palabras que pueden venir después de Reforma, como Reforma Ley y Reforma Económico. Ocurre que como muchos aspectos del procesamiento del lenguaje natural, el contexto es muy importante. Y para las colocaciones, el contexto es muy importante. En el caso de colocaciones, el contexto será un documento en forma de lista de palabras. [9]

Los dos tipos más comunes de colocación son los bigramas y los trigramas. Los bigramas son dos palabras adyacentes, como 'déficit corriente', 'fondo monetario' o 'cartera laboral'. Los trigramas son tres palabras adyacentes, como "fuera del negocio" o "Cristina de Kirchner".

En este trabajo solo se otorgó importancia a la collocations de la clase : “ bigramas”.
Algunos usos para la identificación de colocaciones son:

a) Extracción de palabras clave: identificar las palabras clave más relevantes en los documentos para evaluar de qué aspectos se habla más

b) Bigramas/trigramas se pueden concatenar (por ejemplo, Cristina Fernandez-> Cristina_Fernandez) y contarse como una sola palabra para mejorar el análisis de conocimientos, el modelado de temas y crear funciones más significativas para modelos predictivos en problemas de PNL.

Descubrir colocaciones en esta lista de palabras significa que encontraremos frases comunes que ocurren con frecuencia a lo largo del texto. Para ello a partir de la lista de palabras procesadas anteriormente se crea una nueva lista donde produciremos BigramCollocationFinder, que podemos usar para encontrar bigramas, que son pares de palabras. Estos bigramas se encuentran utilizando funciones de medición de asociación en el paquete nltk.metrics.

De todos los bigramas realizados en base a probabilidades, se optó por las primeras 250 primera palabras de la lista collocations y unirlas con la lista original sacando palabras usadas para generar collocations.

La siguiente tabla data de un pedazo de lista dada en una tabla con la ocurrencia de bigramas en el documento del periodo de 1995.

Ocurrencia	número de ocurrencias
cerrar_compartir :	107
nota_amigo :	107
email_amigo :	107
nombre_email :	107

Si bien se usó las frecuencias para sacar bigramas. Cabe resaltar que hay más métodos.

Exploramos tres métodos para filtrar las colocaciones más significativas:

1. Conteo de frecuencias de palabras adyacentes con filtros:
2. Información mutua puntual
3. Hypothesis Testing

La puntuación Pointwise Mutual Information (PMI) para bigramas es:

$$PMI(w^1, w^2) = log_2 \frac{P(w^1, w^2)}{P(w^1)P(w^2)}$$ [11]

La intuición principal es que mide cuánto más probable es que las palabras co-ocurran que si fueran independientes. Sin embargo, es muy sensible a combinaciones raras de palabras. Por ejemplo, si aparece un bigrama aleatorio 'abc xyz', y ni 'abc' ni 'xyz' aparecen en ninguna otra parte del texto, 'abc xyz' se identificará como un bigrama muy significativo cuando podría ser simplemente un error ortográfico aleatorio o una frase demasiado rara para generalizar como un bigrama.

El test de hipótesis es un proceso para determinar la validez de una aseveración hecha sobre la población basándose en evidencia muestral
Especifica cuándo se puede aceptar o rechazar una afirmación sobre una población dependiendo de la evidencia de los datos.

Una prueba de hipótesis examina dos hipótesis opuestas sobre una población: la hipótesis nula y la hipótesis alternativa. La hipótesis nula es la afirmación que se está comprobando. Normalmente la hipótesis nula es una afirmación de "sin efecto" o "sin diferencia". La hipótesis alternativa es la afirmación que se desea ser capaz de concluir que es verdadera basándose en la evidencia proporcionada por los datos de la muestra.

Basándose en los datos de la muestra, la prueba determina cuándo rechazar la hipótesis nula. Se utiliza un p-valor, para realizar esa determinación. Si el p-valor es menos que el nivel de significación (conocido como α o alfa), entonces se puede

rechazar la hipótesis nula.

5. Algoritmo de aprendizaje automático.

El aprendizaje automático (ML en inglés) es un tipo de algoritmo que se puede mejorar automáticamente a sí mismo basado en la experiencia, sin un programador. El algoritmo gana experiencia al procesar más datos, y luego aprendiendo de los errores cometidos y modificándose.

Una parte del ML es el procesamiento de lenguaje natural, cuyas raíces contienen al algoritmo de LDA.

5.1 Tipos de aprendizaje automático

Hay muchas variedades de técnicas de aprendizaje automático, nombraremos sólo tres enfoques generales:

Aprendizaje reforzado: El algoritmo ejecuta las acciones que serán las más recompensadas. Estos algoritmos son usados por ejemplo en IA para juegos, o en robots de navegación como así en coches automáticos.

Aprendizaje automático no supervisado: El aprendizaje no supervisado parte de datos no etiquetados previamente. El algoritmo encuentra patrones en los datos no etiquetados al agrupar e identificar similitudes entre ellos. Los usos populares incluyen sistemas de recomendación y publicidad enfocada [12].
Por ejemplo el modelado de tópicos (LDA) cae dentro de esta categoría. Otro ejemplo es PCA y k-means.

Aprendizaje automático supervisado: El aprendizaje supervisado supone que partimos de un conjunto de datos etiquetado previamente, es decir, conocemos el valor del target para el conjunto de datos que disponemos.
El algoritmo analiza los datos etiquetados y aprende a asignar etiquetas de salida a datos de entrada. A menudo se usa para clasificación y predicción.[12]

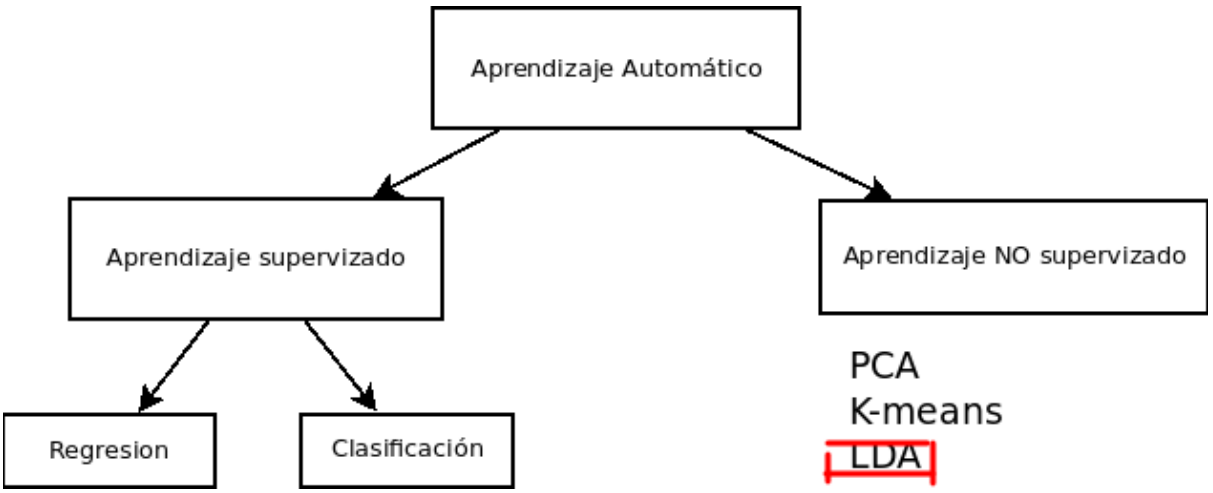


imagen 5.1

5.2 LDA.

Los fundamentos teóricos de LDA se basan en la explotación de los conceptos de intercambiabilidad con el teorema de Finetti (1990)[13].

De acuerdo con el teorema de De Finetti, las observaciones intercambiables son condicionalmente independientes en relación con una variable latente, por lo que se puede asignar una probabilidad epistémica a la variable latente.

Esencialmente, utilizando el teorema de De Finetti, es posible capturar una estructura estadística significativa dentro del documento a través de la distribución de la mezcla.

El topic model o modelado de temas se le considera parte de aprendizaje automático no supervisado que consiente el procesamiento de grandes colecciones de datos.

El objetivo del modelado de temas es hallar variables latentes que gobiernan la semántica de un documento, estas variables latentes representan temas abstractos.

Actualmente, la técnica más popular para el modelado de temas es la Latent Dirichlet Allocation (LDA) [14], y este modelo se puede usar de manera efectiva en una variedad de tipos de documentos, como colecciones de artículos de noticias (de nuestro interés), como así en documentos de economía, publicaciones en twitter o redes sociales.

Representación del modelo LDA:

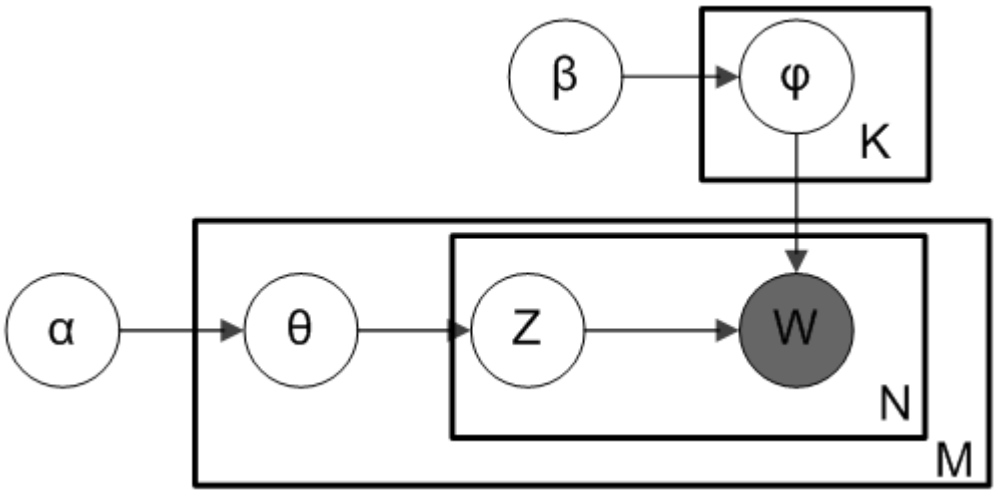


imagen 5.2

K: Número de temas
N: Número de palabras en el documento
M: Número de documentos a analizar
α : Distribución de temas por documento

Un α alto significa que es probable que cada documento contenga una combinación de la mayoría de los temas y no solo un tema específico.

Un α bajo significa que es más probable que un documento esté representado por solo una parte del tema.

β : Distribución de palabras por tema.
φ(k): Distribución de palabras para el tema k (Probabilidad de tema por palabra).
Θ(i): Distribución de temas para el documento I (Probabilidad de temas por documento) .
Z(i,j): Asignación de temas para w(i,j) .
w(i,j): j-ésima palabra en i-ésimo documento.

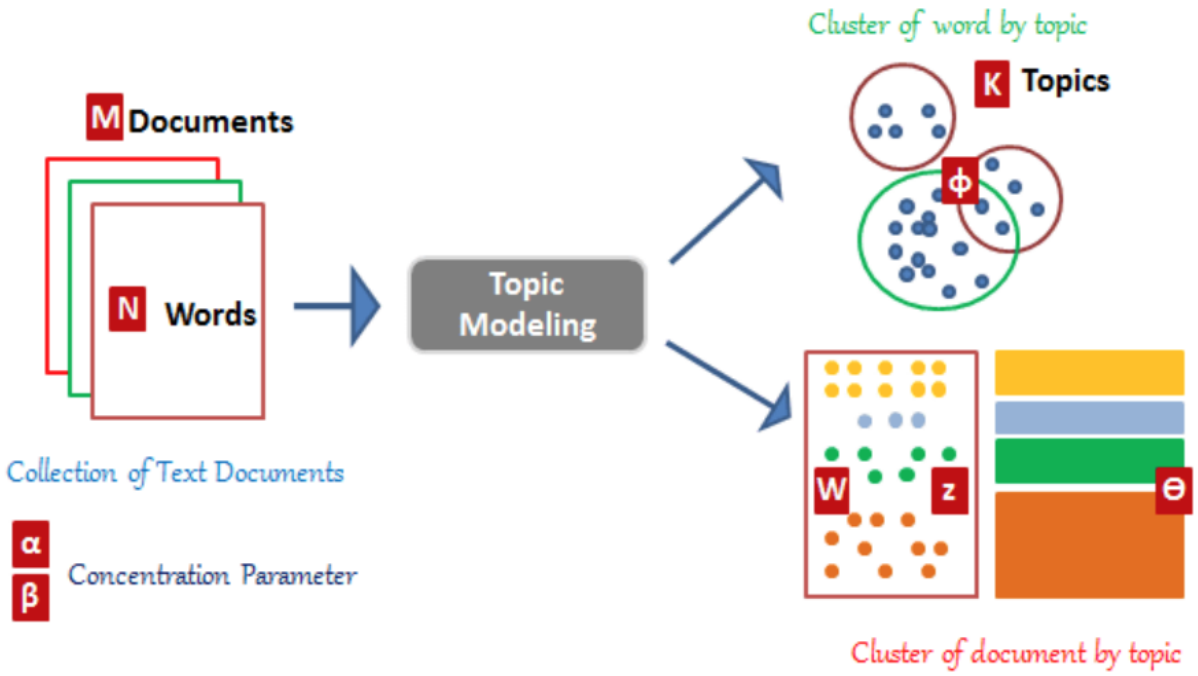


imagen 5.3

Entrada LDA:
1. M documentos.
2. Cada uno de estos documentos tiene N de palabras .
3. Estas palabras necesitan pasar por LDA.

Salida LDA:
K número de temas (grupo de palabras) .
Distribución Φ (distribución de documento a tema).

5.3 Historia de LDA

En esta sección se verá una breve historia y una comprensión teórica de las técnicas fundamentales que dieron a LDA, lo que hoy es.

En la década de 1980, el campo de la recuperación de información produjo un esquema de representación de texto llamado frecuencia de término-frecuencia de documento inversa o del inglés term frequency–inverse document frequency (tf-idf) para aplicar una estadística numérica a una palabra que sería representativa de su importancia dentro de un documento .[15]

La vectorización Tf-idf es donde una frecuencia de término normalizada (recuento del número de ocurrencias de un término dentro de un documento) se compara con la frecuencia de documento inversa normalizada (recuento del número de ocurrencias de un término dentro de un corpus) en una escala logarítmica . Esto da como resultado una matriz de pesos término por documento que lamentablemente es muy escasa, ruidosa y redundante. El proceso para calcular el vector tf-idf para cualquier palabra en un documento se puede representar mediante la siguiente ecuación: [16]

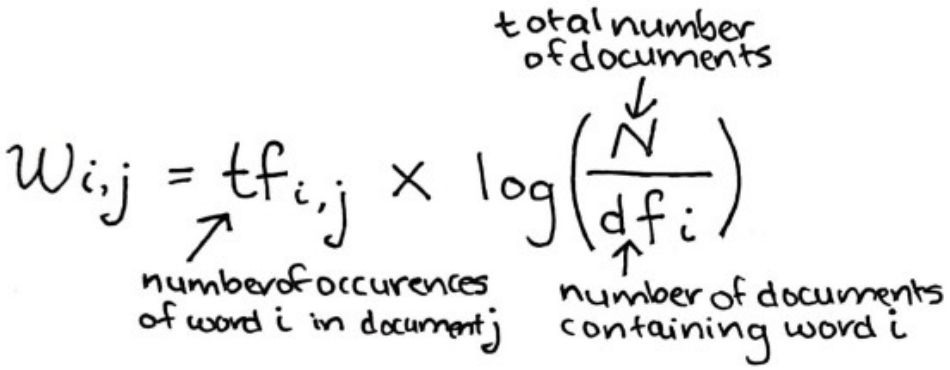


Imagen 5.4 . Ecuación para calcular un vector tf-idf para un término $w_{i,□}$. Imagen por Autor.

Esta herramienta fue pilar fundamental para años posteriores desarrollar LDA.

LDA tiene raíces en la biología evolutiva; allá por el año 2000 los investigadores desarrollaron este modelo para el estudio de la genética de poblaciones. Unos años más tarde, Blei et al., 2003, [14] aplicó LDA al campo del aprendizaje automático. LDA es un modelo probabilístico generativo, específicamente es un modelo bayesiano jerárquico de tres niveles.

Se puede pensar en LDA como una versión bayesiana de pLSI que permite una mejor generalización.

5.4 Matemáticas detrás de LDA.

Un modelo de lenguaje estadístico normal asume que puede generar un documento mediante el muestreo de una distribución de probabilidad sobre

palabras, es decir, para cada palabra en nuestro vocabulario hay una probabilidad asociada de que esa palabra aparezca.

LDA agrega una capa de complejidad sobre este mismo arreglo. Asume una lista de temas, k. Cada documento m es una distribución de probabilidad sobre estos k temas, y cada tema es una distribución de probabilidad sobre todos los diferentes términos de nuestro vocabulario.

Es decir que cada palabra tiene varias probabilidades de aparecer en cada tema. La fórmula de probabilidad completa que genera un documento se encuentra en la imagen 5.7 a continuación. Si desglosamos esto, en el lado derecho tenemos tres sumas de productos:

- Distribución de Dirichlet de temas sobre términos: (corresponde a la imagen 1.5) para cada tema i entre K temas, distribución de probabilidad de palabras para i.
- Distribución de Dirichlet de documentos sobre temas: (corresponde a la imagen 5.5) para cada documento j en nuestro corpus de tamaño M, ¿cuál es la distribución de probabilidad de temas para j. Probabilidad de que aparezca un tema dado un documento X ?
- Probabilidad de que aparezca una palabra dado un tema: (correspondiente a los dos rectángulos en la imagen 5.6) ¿Qué tan probable es que ciertos temas, Z, aparezcan en este documento y luego qué tan probable es que ciertas palabras, W, aparecen dados esos temas.

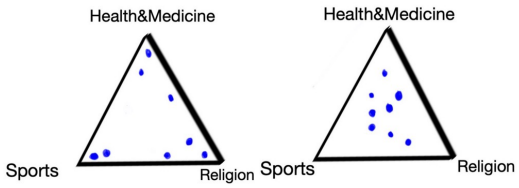


imagen 5.5

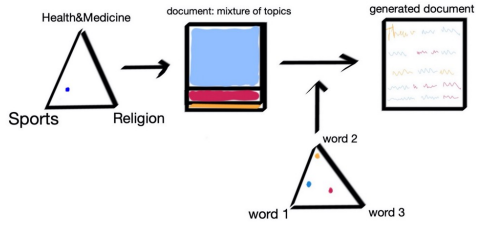


imagen 5.6

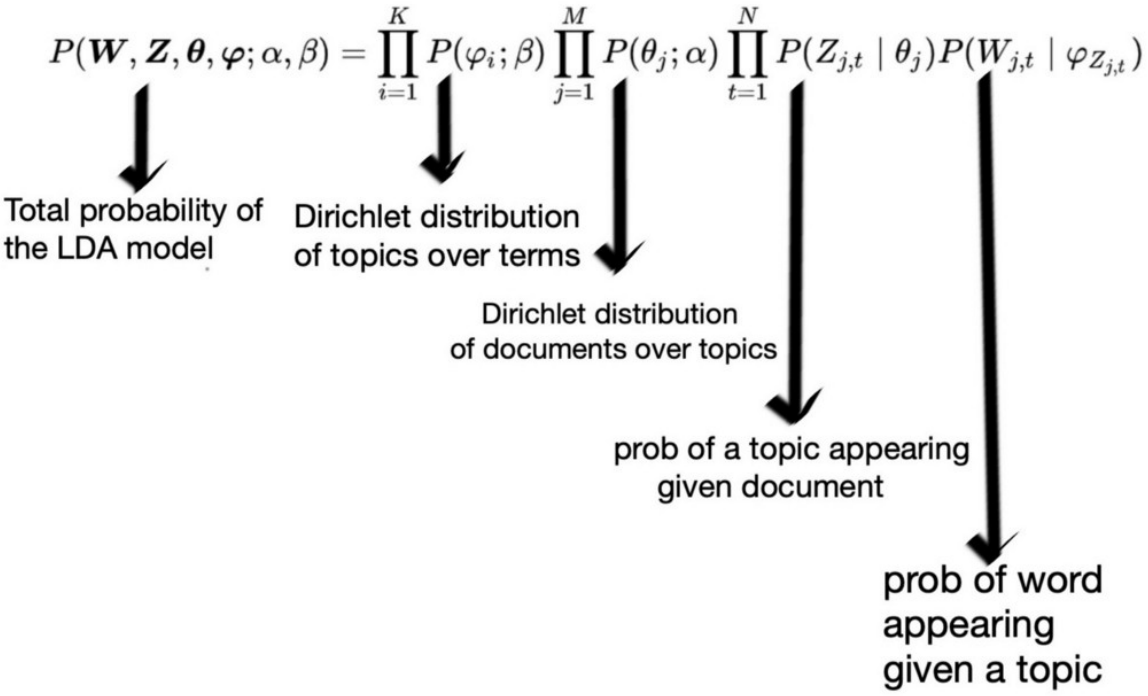


imagen 5.7

5.5 Estructura del modelo LDA

La innovación de LDA está en el uso de anteriores versiones como se mencionó en su historia para las distribuciones documento-tema y término-tema. Esta innovación lo que permite es la inferencia bayesiana sobre un modelo jerárquico de tres niveles (K, N, M), siendo la tercera capa la característica distintiva en comparación con un modelo de agrupamiento multinomial simple de Dirichlet. Con respecto a la inferencia bayesiana, la notación de capas es un método intuitivo para representar gráficamente variables que se repiten; se utiliza una "capas" para representar réplicas y los bordes indican dependencias condicionales. Como se ve en el siguiente diagrama la capa exterior representa documentos y la capa interior representa opciones repetidas de temas y palabras dentro de un documento.

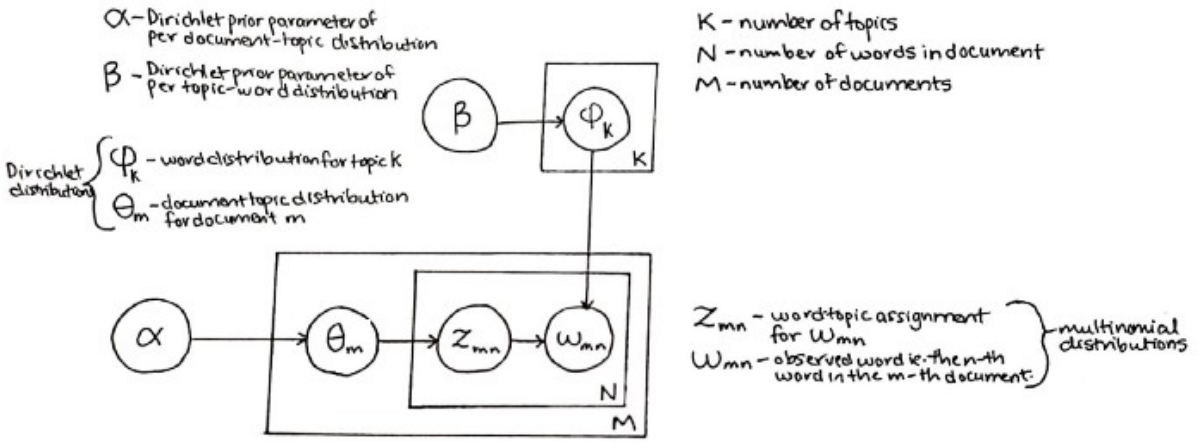


Imagen 5. 8. estructura de LDA

Las distribuciones de Dirichlet es una familia de distribuciones de probabilidad continuas multivariada, permiten el muestreo de distribución de probabilidad en el que todos los números suman 1, y estos números representan probabilidades sobre K categorías distintas.

Una distribución de Dirichlet de dimensión K tiene k parámetros y representa la incertidumbre como una distribución de probabilidad. Los parámetros previos de Dirichlet α y β son parámetros a nivel de corpus que se muestrean una vez en el proceso de generación de un corpus, y el parámetro θ□ es una variable a nivel de documento que se muestrea una vez por documento. Las variables z□□ y w□□ son variables de nivel de palabra que se muestrean una vez para cada palabra en cada documento. Por último, φ□ representa la distribución de probabilidad de palabras para un tema k

5.6 Procedimiento y probabilidad de corpus

Con el modelo LDA, los documentos se representan como mezclas aleatorias sobre temas latentes, donde cada tema se caracteriza por una distribución entre palabras. Este proceso generativo para cada documento dentro de un corpus se puede escribir simplemente de la siguiente manera:

Inicializa cada parámetro de tema β□ ~ Dirichlet (φ), para k ϵ {1…K}

Para cada documento:

1) Elige la distribución del tema θ□ ~ Dirichlet (α)
2) Para cada una de las N palabras w□ :
   a) Elige un tema z□□ ~ Multinomial(θ□)
   b) Elige una palabra w□ ~ Multinomial(β□).

La probabilidad de un corpus (D) es el resultado de tomar el producto de las probabilidades marginales de un solo documento, y la distribución marginal para un solo documento se obtiene integrando sobre θ y sumando sobre z temas.

$$P(D|\alpha,\beta) = \prod_{m=1}^{M} \int P(\theta_m|\alpha)\left(\prod_{n=1}^{N_m} \sum_{z_{mn}} P(z_{mn}|\theta_m)P(w_{mn}|z_{mn},\beta)\right) m\theta_m$$

Image 5.9 A partir del modelo LDA, la probabilidad de un cuerpo D se modela mediante la ecuación de probabilidad P(D|α,β). Imagen por Autor.

Comprender la importancia de las distribuciones de Dirichlet en LDA requiere una comprensión del Teorema de Bayes que establece que, si el anterior tiene una distribución de Dirichlet (α, β) y la probabilidad tiene una distribución multinomial (z□□, w□□), el posterior tendrá una distribución de Dirichlet y puede por lo tanto ser computado. Sin embargo, la distribución posterior es intratable para la inferencia exacta, y hay varios algoritmos de inferencia de aproximación, como la aproximación de Laplace, la cadena de Markov Monte Carlo (por ejemplo, el muestreo de Gibbs) .

5.7 Implementando el modelo LDA

En la asignación de Dirichlet latente (LDA) sus usos incluyen procesamiento de lenguaje natural (NLP) y modelado de temas, entre otros

Se requiere un conjunto de datos razonablemente grande para construir un modelo LDA. El tamaño mínimo necesario depende de las características y longitud media de los documentos. En general, cuanto mayor sea el conjunto de datos, mejores serán los resultados, debido a un aumento en las observaciones. Por ejemplo, cuando se trabaja con artículos de noticias, el número mínimo de artículos necesarios para el modelado es de 3400; sin embargo, los resultados pueden mejorar cuando el conjunto de datos se le incluye más artículos.

Las siguientes cuatro secciones cubrirán los detalles necesarios para implementar con éxito LDA para el modelado de temas de datos de texto en artículos de diarios.

La primera sección proporciona una descripción de hiper parámetros seleccionados del modelo LDA, incluidos los anteriores de Dirichlet.

En segundo lugar, es importante considerar el problema de la inferencia; específicamente, nos enfocaremos en la solución de un algoritmo Bayesiano.

Un ejemplo del modelo de temas estimado por LDA consta de dos tablas (matrices). La primera tabla describe la probabilidad de seleccionar una parte en particular al muestrear un tema en particular (categoría). La segunda tabla describe la posibilidad de seleccionar un tema en particular al muestrear un documento.

	Gobernador(G)	País(P)	Sociedad(S)
Documento 0	10	0	0
Documento 1	0	10	0
Documento 2	0	0	10
Documento 3	10	10	10

Tabla 5.1

Documento 0	G G G G G G G G G G
Documento 1	P P P P P P P P P P
Documento 2	S S S S S S S S S S
Documento 3	G G G G G G G G G P P P P P P P P P P S S S S S S S S S S

Tabla 5.2

Tomemos la Tabla 5.1 y la Tabla 5.2 de arriba, por ejemplo, cuatro secuencias de palabras (los 'compuestos') y tres tipos de palabras (las 'partes'). Los primeros tres documentos tienen diez de un solo tipo de palabras, mientras que el último

documento tiene diez de cada tipo.

	Topic 0	Topic 1	Topic 2
Gobierno	0.00	0.00	0.99
País	0.99	0.00	0.00
Sociedad	0.00	0.99	0.00

Tabla 5.3

	Topic 0	Topic 1	Topic 2
Doc 0	0.03	0.03	0.939
Doc 1	0.939	0.03	0.03
Doc 2	0.03	0.939	0.03
Doc 3	0.3	0.3	0.3

Tabla 5.4

Después de ejecutar nuestros compuestos de 5.3 palabras a través de LDA, terminamos con el modelo de tema probabilístico que vemos arriba.

La tabla 3 tiene 'palabra vs tópicos', y la tabla 5.4 muestra 'documentos vs tópicos'. Entonces cada columna de la tabla 5.3 y cada fila de la tabla 5.4 suman uno (nos permitimos algún truncamiento y pérdida de precisión). Entonces, si tuviera que sacar una palabra de una bolsa, el Tema 0, casi seguro obtendría la palabra Gobierno. Si tomáramos una muestra del Documento 3, hay una probabilidad igual (o "uniforme") de obtener el Tema 0, 1 o 2.

5.8 Hiper Parámetros del modelo LDA

Hay varias bibliotecas de Python con módulos LDA. Se empezó este trabajo usando una librería llamada Gensim. No obstante (personalmente) se decidió usar otra librería de python llamada Sklearn debido a que Sklearn trabaja con cython y es 3 veces más rápido que gensim.[17]

Gensim es una biblioteca de código abierto para el modelado de temas no supervisados y el procesamiento del lenguaje natural, utilizando el aprendizaje automático estadístico moderno. Está diseñado para manejar grandes colecciones de texto utilizando transmisión de datos y algoritmos en línea incrementales, lo que lo diferencia de la mayoría de los otros paquetes de software de aprendizaje automático que se enfocan solo en el procesamiento en memoria.

Al implementar LDA con gensim, es necesario ajustar los hiper parámetros, específicamente el número de temas (k), el número de características (V, es decir, tamaño de vocabulario fijo) y los parámetros previos de Dirichlet α y β. Es posible ajustar otros parámetros como el número de iteraciones, el método de aprendizaje (por lotes o en línea), la compensación de aprendizaje, la tolerancia a la perplejidad y otros, pero no es necesario para el objetivo de este trabajo.

La biblioteca sklearn contiene muchas herramientas eficientes para el aprendizaje automático y el modelado estadístico, incluidas la clasificación, la regresión, el agrupamiento y la reducción de la dimensionalidad. Así como gensim, Skit learn o sklearn se admite el ajuste de hiper parámetros como los ya nombrados: el número de temas (k), el número de características y los parámetros de Dirichlet α y β.

5.9 Número de temas (k)

El hiper parámetro más importante es el número de temas, cuya elección depende de las características y el tamaño del conjunto de datos. Por ejemplo, cuanto mayor sea el conjunto de datos, mayor será la cantidad de temas, solo si el conjunto de datos es representativo de una colección diversa. Sin embargo, una colección de unos pocos miles de artículos científicos sobre un tema en particular podría no contener más temas si se agregan varios miles de artículos similares al conjunto de datos inicial.

El enfoque heurístico es aprovechar el conocimiento del contenido del conjunto de datos para estimar un objetivo probable y realizar ajustes basados en la evaluación del modelo y visualizaciones basadas en la reducción de la dimensionalidad. El número óptimo de temas se puede determinar calculando las puntuaciones de coherencia de temas en un rango de números de temas y trazando la tendencia de coherencia de temas resultante (HDP de esta técnica se hablará más adelante). Sin embargo, esta es una tarea computacionalmente costosa, por lo tanto, es mucho más eficiente estimar heurísticamente un valor inicial para k y usar técnicas de evaluación de modelos para guiar empíricamente los ajustes.

5.9.1 Número de características (V)

El mismo enfoque se puede utilizar para elegir el número de funciones, lo que equivale a establecer un tamaño fijo para el vocabulario. Cuanto mayor sea el

número de funciones, más tiempo tardará en entrenarse el modelo LDA; sin embargo, se necesita un vocabulario suficientemente grande para capturar las palabras más importantes para la agrupación de temas. Por lo general, establecer el número de funciones en 10.000 es un buen punto de partida para la mayoría de los modelos. Este valor debe ajustarse según el tamaño del conjunto de datos y la cantidad de diversidad de las palabras en la colección.

5.10 Los hiper parámetros α y β

La mayoría de los modelos LDA asumen una distribución simétrica, y los parámetros α y β actúan como antes del cálculo posterior. Esta suposición de simetría significaría que cada tema se distribuye uniformemente a lo largo de un documento, mientras que para una distribución asimétrica (medida por la asimetría) ciertos temas se verían favorecidos sobre otros. Como parámetros de concentración previa de Dirichlet, α y β, son representativos de la densidad de documento-tema y la densidad de tema-palabra, respectivamente.

El parámetro α especificará creencias previas sobre la escasez y uniformidad de los temas; visualizado como una matriz, cada fila es un documento y cada columna es un tema (como vimos en la sección 5.7 tabla 4). Con un valor alto de α, se supone que los documentos contienen más temas. Esencialmente, esto significa que es probable que cada documento contenga una combinación de muchos temas y no un solo tema específico. Por el contrario, un valor bajo de α supone que un documento contendrá una mezcla de solo unos pocos o un solo tema. Esto sucede porque a medida que disminuye el valor de α, la dispersión aumenta de tal manera que, cuando se muestrea la distribución, la mayoría de los valores serán cero o cercanos a cero.

Además, si las distribuciones son asimétricas, un valor alto de α da como resultado una distribución de temas más específica por documento. Inicialmente, el parámetro α se puede establecer en un valor de número real dividido por el número de temas, y los resultados deberían revelar una idea de la escasez y simetría de la distribución. Por lo tanto, el enfoque heurístico para elegir un valor de α es estimar la escasez tópica de cada documento en promedio. Los ajustes posteriores deben determinarse mediante la evaluación del modelo y luego probarse empíricamente.

Como se mencionó, el parámetro β representa la densidad de palabra-tema (sección 5.7 tabla 5.3), y es una matriz donde cada fila representa un tema y cada columna representa una palabra. El parámetro β especificará creencias previas sobre la escasez de palabras y la uniformidad dentro de los temas, ajustando el sesgo de que ciertos temas favorecerán ciertas palabras. Con un valor alto de β, se supone que los temas están formados por la mayoría de las palabras del vocabulario de tamaño fijo y esto da como resultado una combinación de palabras más específica para cada tema.

Por el contrario, con un valor bajo de β, un tema puede contener una mezcla de solo

algunas de las palabras del vocabulario de tamaño fijo. Además, si la distribución es asimétrica, un valor alto de β dará como resultado una distribución de palabras más específica. Los temas, sin embargo, serán más similares en términos de palabras contenidas.

6. Elegir el número de temas para LDA.

El modelado en NLP busca encontrar la estructura semántica oculta en los documentos. Son modelos probabilísticos analizan grandes cantidades de texto y agrupar grupos similares de documentos sin supervisión.

Una vez que se aplica el modelado de temas LDA al conjunto de documentos, puede ver las palabras que componen cada tema oculto.

Sin embargo, aquí, nos lleva a la siguiente pregunta: ¿Cómo elegimos el número óptimo de k temas?. De esto trataremos de aclarar en este capítulo.

6.1 HDP

Para entrenar un modelo LDA, debemos proporcionar una cantidad fija de temas en todo el corpus. Mencionaremos sólo tres técnicas que fueron encontrados:

- La primera técnica se debe ejecutar LDA en su corpus con diferentes números de temas y notar a ojo si la distribución de palabras por tema tiene la coherencia deseada.
- Otra forma es usar LSI, para examinar las puntuaciones de coherencia de su modelo LDA.
- Por último y la más importante es correr muchos modelos LDA con diferentes valores de tema, luego ver cómo funcionan en el entrenamiento del modelo de clasificación supervisada. Esto es específico para nuestro objetivo, ya que se necesita observar si las distribuciones de temas tienen valor predictivo. Esta técnica es la que se usó para este trabajo.

La primera técnica no se ve útil debido a que considerar el número óptimo de cada corpus armado , se necesita mucho tiempo esfuerzo y además que es una mera percepción personal.
Para la segunda técnica, sin embargo no es lo suficientemente confiable.
El último enfoque , sin embargo, es totalmente razonable para nuestro objetivo.

Gensim también proporciona una clase de proceso jerárquico de Dirichlet (HDP). HDP es similar a LDA, excepto que busca aprender la cantidad correcta de temas de los datos; es decir, no necesita proporcionar un número fijo de temas.
Al usar HDP en los corpus, en todos arrojaron 20 tópicos, por lo que se tomó la decisión de usar este número para el número de temas en todas las ejecuciones de LDA.
No obstante esto se utilizó en etapas iniciales con gensim, al cambiar al algoritmo de sklearn se hicieron ajustes en los parámetros y no se utilizó HDP. Sin embargo creemos beneficioso dar un ejemplo de HDP debido a que es una excelente herramienta para obtener el número óptimo de tópicos.
Tomaremos un ejemplo que se encontró es intuitivo y brillante que explica Edwin Chen en para ver el funcionamiento de HDP:

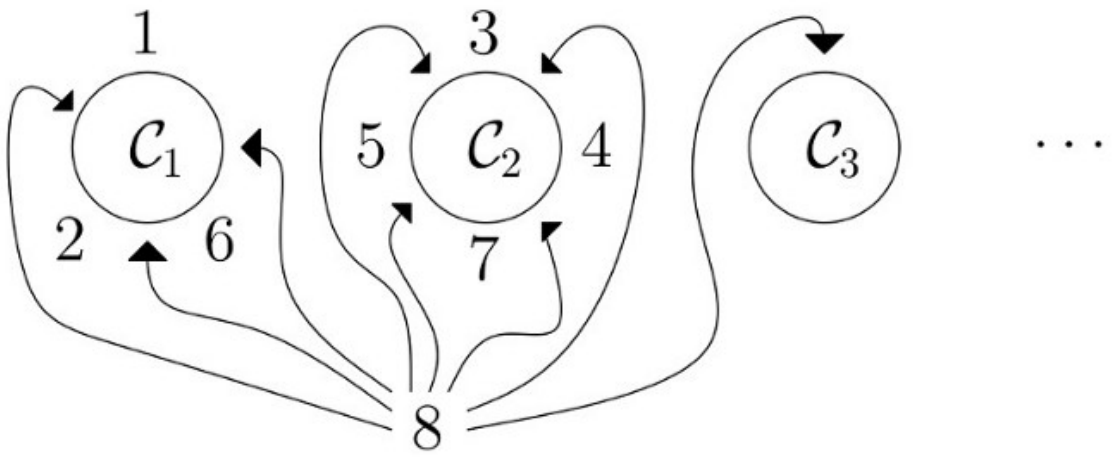


imagen descargada de[19]

El restaurante chino:
Necesitamos asignar 8 a un tópico. Hay una probabilidad de 3 / 8 de que 8 salga del tema C1, una probabilidad de 4 / 8 de que 8 salga del tema C2, una probabilidad de 1 / 8 de que 8 salga del tema C3. De esta manera se descubre una serie de temas. Entonces, cuanto más grande es un grupo, más probable es que alguien se una a ese grupo.[19]

6.2. HLDA

Otro algoritmo que se probó con gensim es HLDA. El modelo “jerárquico LDA” (hLDA) amplía LDA para inferir una jerarquía de temas a partir de un corpus de documentos. Estábamos interesados en ver si podíamos usar esta técnica para organizar automáticamente el número de tópicos.

hLDA nos permite asumir que nuestros temas están organizados en una estructura similar a un árbol donde el árbol tiene L niveles y cada nodo es un tema.

Con LDA, elegimos temas utilizando un modelo de mezcla con K proporciones de mezcla aleatorias (donde K es el número de temas posibles), indicado por el vector θ K-dimensional. Con hLDA, en su lugar, elegimos una ruta en el árbol de nivel L desde la raíz hasta la hoja, elegimos un vector θ de proporciones de tema de una distribución de Dirichlet de dimensiones L, luego usamos una mezcla de los temas desde la raíz hasta la hoja para generar las palabras que componen cada documento, utilizando proporciones de mezcla θ.

Un problema con este enfoque es que hay muchas estructuras de árbol posibles entre las que podemos elegir, incluso cuando fijamos el número de niveles de árbol en L, porque no hemos fijado el factor de ramificación (el número de hijos en cada nodo) en cualquiera de los niveles del árbol.

No entraremos en detalles porque en este trabajo no logró buenos resultados con esta técnica.

```
HierarchicalLDA sampling

.................................................. 50
topic=0 level=0 (documents=13): reforma, laboral, proyecto, gobierno, menem,
    topic=1 level=1 (documents=7): obra, plan, inversion, propuesta, autopista,
        topic=2 level=2 (documents=7): fondo, economia, fmi, crisis, mercado,
    topic=3 level=1 (documents=6): ley, recinto, oficialismo, pj, quorum,
        topic=6 level=2 (documents=6): diputado, bloque, legislador, alianza, decreto,
.................................................. 100
topic=0 level=0 (documents=13): reforma, laboral, proyecto, gobierno, presidente,
    topic=1 level=1 (documents=8): obra, plan, sector, inversion, infraestructura,
        topic=2 level=2 (documents=8): fmi, fondo, economia, crisis, roque,
    topic=3 level=1 (documents=5): ley, quorum, recinto, oficialismo, alianza,
        topic=6 level=2 (documents=5): bloque, diputado, decreto, legislador, pj,
.................................................. 150
topic=0 level=0 (documents=13): reforma, laboral, proyecto, presidente, menem,
    topic=1 level=1 (documents=8): obra, plan, gobierno, sector, infraestructura,
        topic=2 level=2 (documents=8): fondo, fmi, economia, mercado, capital,
    topic=3 level=1 (documents=5): ley, legislador, recinto, quorum, diputado,
        topic=6 level=2 (documents=5): bloque, articulo, ejecutivo, peronista, norma,
                                                   200
```

7. Sklearn y su implementación.

7.1 Matriz documento-palabra

El algoritmo del modelo de tema LDA requiere una matriz de palabras del documento como entrada principal.

Una matriz document-words matemáticamente describe la frecuencia de las palabras que aparecen en una colección de documentos.

Al crear un conjunto de datos de términos que aparecen en un corpus de documentos, la matriz documento-término contiene filas correspondientes a los documentos y columnas correspondientes a los términos. Cada celda i j, entonces, es el número de veces que aparece la palabra j en el documento i. Como tal, cada fila es un vector de conteo de términos que representa el contenido del documento correspondiente a esa fila. Por ejemplo, se tiene los siguientes dos documentos:

Doc1 = “El presidente Alberto”
Doc2 = “Trabajo de presidente”
Doc3 = “Trabajo de Alberto”

	El	presidente	Alberto	de	trabajo
Doc1	1	1	1	0	0
Doc2	0	1	0	0	1
Doc3	0	0	1	0	1

que muestra qué documentos contienen qué términos y cuántas veces aparecen. Aquí, nos enfrentamos a representar un documento sólo como una lista de recuento de tokens, la matriz de término de documento incluye todos los términos del corpus (es decir, el vocabulario del corpus), razón por la cual hay recuentos cero para los términos del corpus que no también ocurren en un documento específico. Para eliminar estos ceros, y no ocupar espacio de memoria se hablará en la siguiente sección.
Para hacer esta matriz de documento-palabra usamos de sklearn:

```
vectorizer.fit_transform(lista_de_articulos)
```

7.2 Checkeo de sparsicity

En análisis numérico y computación científica, una matriz dispersa o arreglo disperso es una matriz en la que la mayoría de los elementos son cero. Y nuestro objetivo es reducir dicha dimensionalidad por cuestión de memoria, dado que al tener una matriz documento-palabra se aloja en ella un montón de ceros.
Para ello sklearn lo resuelve en una función

```
"matrix.to_dense"
```

7.3 Construyendo el modelo de LDA con sklearn

Se construyeron distintos modelos de LDA para observar su precisión en el estudio de las agrupaciones de palabras.
La estructura del modelo es la siguiente:

```
LatentDirichletAllocation(n_components=        number topic, # Number of topics
                          doc_topic_prior =     alpha,          # alpha
                          topic_word_prior=     beta,           # beta
                          max_iter = iterations,             # Max learning iterations
                          learning_method='online',
                          random_state= seed,         # Random state
                          batch_size=batch,           # n docs in each learning iter
                          evaluate_every = -1,       # compute perplexity every
                                                        #n iters, default: Don't
                          n_jobs = -1              # Use all available CPU
                         )
```

7.3.1 Pruebas en alfa y beta

No entraremos en detalle de la importancia de estos hiper parámetros, dado que, ya lo mencionamos.

Se decidió armar tres grupos de alfa y beta:

- Para un nivel bajo: α=0.2 y β=0.1.

- Para un nivel medio α=0.6 y β=0.4.
- Por último para un nivel alto cercano a uno tal como α=0.99 y β=0.9.

En cada construcción del modelo se realizaron 6, 10, 20, 40 y 60 tópicos.

7.3.2 Rendimiento del modelo con perplexity y log-likelihood.

Un modelo con mayor probabilidad logarítmica y menor perplejidad (exp(-1. * probabilidad logarítmica por palabra)) se considera bueno.
Sin embargo la perplejidad podría no ser la mejor medida para evaluar modelos de temas porque no considera el contexto y las asociaciones semánticas entre palabras.

7.3.3 Usando Grid Search para encontrar el mejor modelo LDA.

El parámetro de ajuste más importante para los modelos LDA es n_components (número de temas). Como así también learning_decay (que controla la tasa de aprendizaje). Además de estos dos hiper parámetros, otros posibles parámetros de búsqueda podrían ser learning_offset (reducir el peso de las primeras iteraciones) y max_iter.
La búsqueda de cuadrícula construye múltiples modelos LDA para todas las combinaciones posibles de valores de parámetro.
Se observa en el apartado 13.1 nuestra grid search para la búsqueda óptima de parámetros.

8. Visualizando el modelo LDA.

8.1 pyLDAvis.

PyLDAvis es una biblioteca de Python de código abierto que ayuda a analizar y crear una visualización interactiva de los clústeres creados por LDA.[25]

El modelado de temas es una parte de Machine Learning donde el modelo automatizado analiza los datos de texto y crea grupos de palabras a través de un conjunto de datos o una combinación de documentos. (Ver capítulo 1).

El modelado de temas funciona para descubrir los temas en el texto y descubrir los patrones ocultos entre las palabras relacionadas con esos temas (se explicó en detalle en el apartado que explica a LDA). Al usar el modelado de temas, podemos crear grupos de documentos que son relevantes, por ejemplo, se puede usar en la industria de contratación para crear grupos de trabajos y solicitantes de empleo que tienen conjuntos de habilidades similares. En el presente trabajo se usó para separar en temas por periodo presidencial y obtener deducciones de qué habla cada tópico. Un ejemplo se observa al ver a varios tópicos separados por épocas. Observamos que en página 12 se menciona al presidente Menem en los periodos de 1995. Y el apellido Fernandez toma su luz en el periodo 2002 a 2008. Mientras que Macri en el 2015 en adelante.

Hay varias formas de obtener los temas del modelo, pero en este apartado reforzaremos estas gráficas sobre LDA-Latent Dirichlet Allocation. Lo importante es que en este capítulo visualizamos los clusters creados por LDA. Véase el capítulo 12.2, se observa en este los distintos resultados de cada tópico.

8.2 Wordcloud de cada tópico.

Word Clouds se convirtió en una técnica de visualización revolucionaria para comprender y determinar patrones y tendencias en evolución. Ya sea para descubrir los movimientos políticos de un país o para analizar las opiniones de los ciudadanos de una región, se puede obtener una representación visual trazando la nube de palabras.

El Word Cloud es una técnica de visualización de textos que se utiliza de forma nativa para visualizar las etiquetas o palabras clave de un texto digitalizado. Para obtener un considerable wordcloud se necesitan muchas palabras en el texto. Estas

palabras clave suelen ser palabras individuales que representan el contexto del texto a partir de la cual se crea la nube de palabras. Para obtener estas palabras claves (véase capítulo 5) , al entrenar un modelo de LDA ocasionó una lista regulada de palabras. Dicha lista representa la una especie de frecuencia
Estas palabras enlistadas se agrupan para formar una nube de palabras. Cada palabra de esta nube tiene un tamaño de letra y un tono de color variables. Así, esta representación ayuda a determinar palabras de prominencia.
Un tamaño de fuente más grande de una palabra representa su prominencia más en relación con otras palabras en el grupo. Word Cloud se puede construir en diferentes formas y tamaños.

La cantidad de palabras juega un papel importante al crear una nube de palabras. Más cantidad de palabras no siempre significa una mejor nube de palabras, ya que se vuelve desordenada y difícil de leer. Una nube de palabras siempre debe ser semánticamente significativa y debe representar para qué está destinada. Aunque hay diferentes formas en que se pueden crear nubes de palabras, pero el tipo más utilizado es mediante el uso de la frecuencia de las palabras en nuestro corpus. Y así, crearemos nuestra nube de palabras usando el tipo de frecuencia.

En el apartado 13.3 se muestra en tablas las frecuencias de cada tópico y a su vez también mostramos la nube de palabras relacionadas a la tabla.

8.3 Lista de palabras.

Una de las partes menos favoritas de este trabajo, fue el manejo de listas de listas o listas de array. Era de esperarse al hacer print en ciertos resultados un error que daba como output poca memoria, inclusive en un servidor proporcionado por la facultad, como así en otros momentos debía de reiniciar el kernel porque no realizaba la tarea dada.
Una solución fue guardar en txt algunas listas útiles y necesarias para pruebas, se verá en el capítulo 10.
Se trabajó con listas de arreglos las cuales cada arreglo representa el word embeddings en fastext, por cada período, por cada artículo. Esto produjo un vector enorme, la cual, la memoria imposibilitó procesar.

9. Word embedding y Fasttext.

9.1 Word embedding.

Antes de adentrarnos en Fasttext, empezaremos con la definición de word embeddings y ejemplos sencillos.

Word Embedding es una técnica de modelado de lenguaje que se utiliza para representar palabras a vectores de números reales. Representa palabras o frases en espacio vectorial con varias dimensiones. Dicha representación tiene propiedades de agrupamiento útiles, ya que agrupa palabras que son semánticamente y sintácticamente similares. Por ejemplo, esperamos que las palabras “Leyes” , “Gobernador” y “Presidente” se encuentren cerca, pero “Rector” y “Gobernador” no se encuentren ya que no existe una fuerte relación entre ellas. Por lo tanto, las palabras se representan como vectores de valores reales, donde cada valor captura una dimensión del significado de la palabra. Esto provoca que palabras semánticamente similares, tengan vectores similares.

Las incrustaciones de palabras (words embeddings) se pueden generar utilizando varios métodos, como redes neuronales, matriz de co-ocurrencia, modelos probabilísticos, etc.

Veamos un ejemplo sencillo con la técnica de matriz de co-ocurrencia. Supongamos que tenemos un texto en el cual se obtuvo el siguiente contexto.

	X: Universidad	Y: País	Z: Ciudad
Rector	1	0	0
Gobernador	0	1	0
Presidente	0	0	1
Leyes	0	1	1

Es considerable pensar que X, Y, Z son contextos y que “Leyes” se acerca más a “Ciudad” con “País” que “Universidad”. Al tratarse de una matriz de conteo se puede pensar como los vectores:

Rector: (1,0,0)
Gobernador: (0,1,0)
Presidente: (0,0,1)
Leyes: (0,1,1)

LLevando estos valores a un sistema cartesiano, obtenemos:

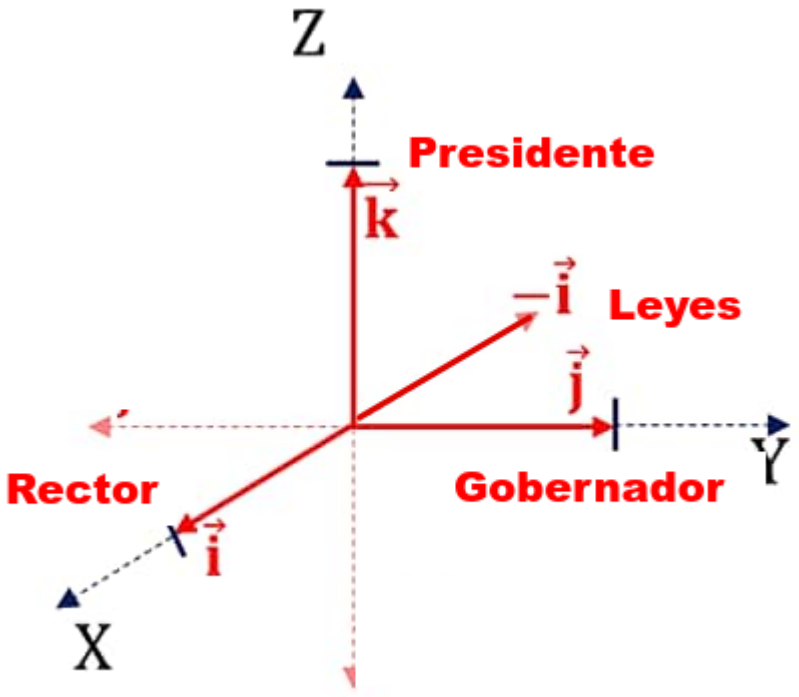


Imagen 9.1

Hay distintas formas de representar palabras en una serie de documentos. Los word embeddings encuentran una representación vectorial compacta.

Su principal característica es que las expresiones relacionadas entre sí (ya sea por contexto, semántica o sintaxis) se encuentren cercanas en el espacio vectorial al cual se proyectan. Por ejemplo, si tenemos las oraciones: “El presidente dictará las leyes“ y “El dirigente discutirá las leyes“. Difieren semánticamente solo por las palabras “dirigente“ y “presidente“. Por lo que, dichas oraciones se utilizan en contextos similares, con lo cual esperaríamos que estén cercanas vectorialmente.

Otro ejemplo muy famoso que deja en evidencia el concepto de analogía en word embeddings, véase Imagen 9.2. Notamos que se identifican por medio de barras de colores la dimensión de los vectores asociados a las palabras king, man, woman, la operación king - man + woman y queen (operación vectorial).

A cada dimensión se le asigna el color de rojo fuerte para valores cercanos al extremo derecho, azul para las del extremo izquierdo, y blanco para las que se encuentren alrededor del 0.

Algo a notar es que man se asemeja a king y woman, por su parte, se acerca a queen. Otra observación a notar es que las franjas rojas y azules que se muestran en la imagen 9.2 , indican qué tan parecidas son en tal dimensión.

Dado que ahora tienen una representación vectorial, se pueden sumar o restar obteniendo nuevos resultados que forman parte del espacio. Este es el caso del vector dado por king - man + woman. No conocemos qué dimensiones exactamente capturan qué rey es utilizado para identificar al monarca de un reino, pero se puede sustraer el carácter masculino del vector man y agregar las de woman esperando que se aproximen a las de queen. Sorprendentemente, de un total de 400000 palabras sobre las cuales se realizó el experimento, reina fue la palabra más cercana. Los detalles del mismo se pueden ver desde (The illustrated Word2Vec, 2019).

Esto refleja el gran potencial que tienen los word embeddings y su importancia para estudiarlos como métodos de codificación. A continuación analizaremos cómo se calculan dichas representaciones y qué algoritmos fueron finalmente utilizados en este trabajo.

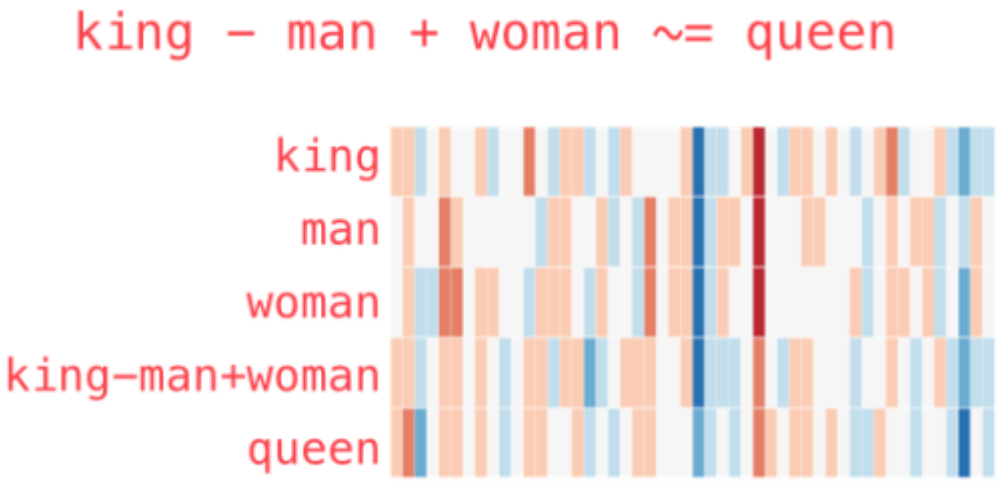


Imagen 9.2. Imagen de un curso “Diplomatura en ciencias de datos” (2021)

9.2 Modelos de lenguaje:

Una de las técnicas de representación es el modelo de Bag Of Words o bolsa de palabras.

9.2.1 Método BOW (Bolsa de palabras):

La representación de textos más extendida es Bag-of-Words [22]. Ofrece resultados razonables en el procesamiento del lenguaje natural. El modelo de Bag of Words toma como entrada un conjunto de textos y genera una representación vectorial de longitud constante. La representación resultante denota
las ocurrencias de cada término a lo largo de los documentos que forman parte del corpus.

1. El primer paso para generar un modelo de bolsa de palabras es tokenizar el documento por palabras (véase capítulo 3.1).

2. Seguidamente se aplica un método de conteo de las palabras y se normaliza el conteo resultante. Para ello, se utiliza el producto de dos estadísticas, TF-IDF (Term Frequency - Inverse Document Frecuency), que en conjunto reflejan la relevancia de la palabra en el documento.

TF-IDF considera como relevantes y asigna un peso mayor a aquellos términos que tienen una ocurrencia alta, pero aparecen en pocos documentos. Esta fórmula nos permite discriminar palabras conocidas como stopwords (véase capítulo 3.2.1), que aparecen muy frecuentemente a lo largo de los documentos, pero que están vacías.

$$T\,f - Idf\,(t, d, D) = tf\,(t, d) * idf\,(t, D)$$

(t: término, d: documento, D: conjunto de documentos del corpus)

Para el conteo de términos (tf) podemos optar por la frecuencia del término (en base a la longitud del documento). Para el cálculo de la inversa de la frecuencia (idf) disponemos de la versión básica. La configuración básica de la fórmula sería la siguiente:

$$T\,f - idf\,(t, d, D) = f\,(t, d) * logN/nt$$

Esta representación es sencilla de construir, pero no refleja toda la riqueza del texto.

Desventajas:

- No se tiene en cuenta el orden de las palabras a la hora de generar la representación. Esto hace que se pierda el aspecto semántico de los términos.

- La gran dimensionalidad del modelo resultante (dispersidad de datos o data sparsity). Esto hace más complejo el cómputo del modelo y su manejo a la hora de ser utilizado.

Bag of Words no deja de ser una representación de la distribución de palabras en el texto desde un punto de vista estadístico, pero sin tener en cuenta el significado de cada palabra o la relación de ésta con su contexto.

9.2.2 CBOW

Los Embedding Vectors [35] surgen de realizar un análisis más específico de los documentos y obtener una representación orientada a las palabras. Se trata de una red de neuronas de dos capas que procesa texto. A partir de un corpus se obtiene un conjunto de vectores que representan los términos.

El modelo de embedding vector presenta dos arquitecturas distintas: Continuous Bag of Words (CBOW) y Skip-gram. Se diferencian entre sí por las features de entrada y el objetivo a predecir (target).

En el modelo de Continuous Bag of Words (CBOW), utilizamos el contexto para inferir el término al que corresponde, y en el método Skip-gram tomamos como origen un término para obtener el contexto.

En el modelo de Continuous Bag of Words inferimos la palabra actual a partir del contexto. Para ello tomamos las palabras anteriores y posteriores a la palabra a predecir. Consiste en un modelo similar a la versión de memoria distribuida de paragraph vector, pero esta vez sin tener en cuenta la representación del párrafo para la predicción de un término del párrafo.

La red neuronal que utilizamos tiene tres capas: una de entrada, una de proyección y otra de salida (véase imagen 10.4). La capa de entrada recoge las palabras del contexto, y éstas pasan a ser proyectadas en la misma posición. Se realiza una media sobre el vector de representación de cada palabra para predecir el término resultante.

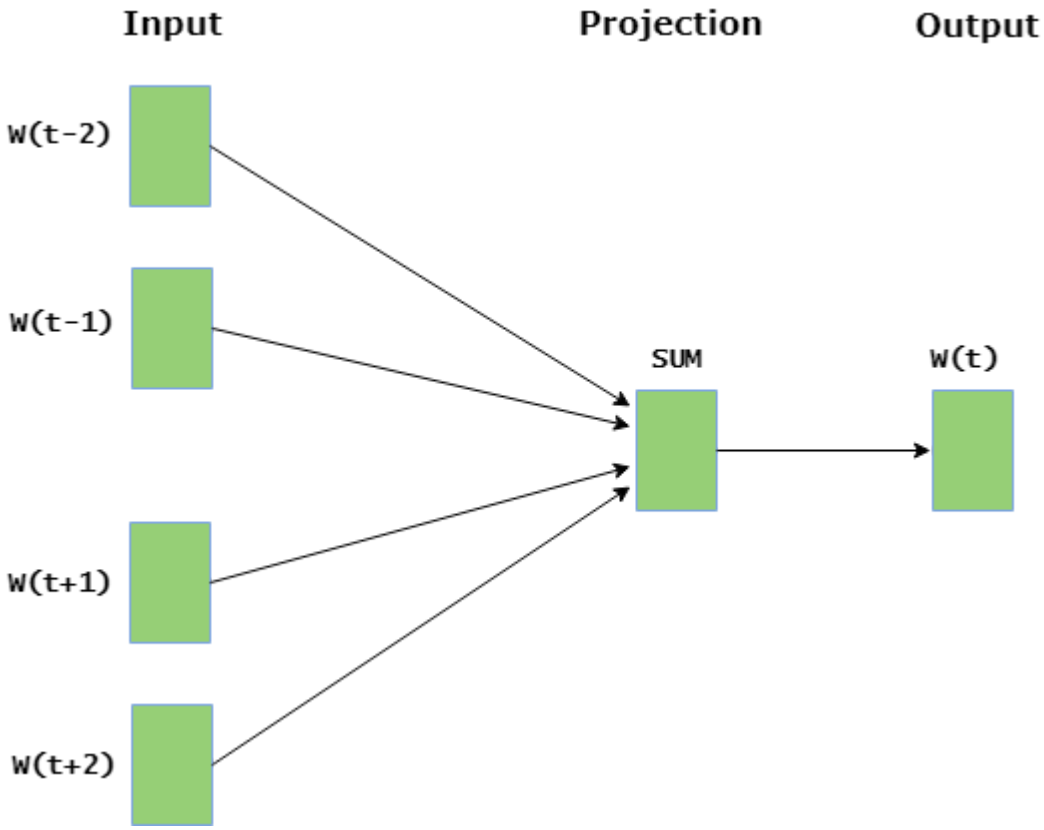


imagen 9.4 arquitectura de CBOW.

La red neuronal que utiliza CBOW es un clasificador softmax. La función de softmax es una versión de la función logística que redimensiona un vector de entrada a otro de dimensión fija y valores normalizados. Para optimizar la implementación de este modelo se utiliza habitualmente la función de hierarchical softmax (softmax jerárquico). La versión de hierarchical hace uso de un árbol binario para la clasificación, que reduce el número de salidas a evaluar (log2(w))), por lo que es menos costoso computacionalmente [35].

Se ha demostrado que CBOW tiene un desempeño mejor con respecto a skipgram sobre datasets más pequeños, ya que con el segundo es necesario disponer de una gran dataset para hacer un buen modelo.

El modelo CBOW predice la palabra actual dadas las palabras de contexto dentro de una ventana específica. Por ejemplo, se requiere predecir la palabra “propuesta” en el siguiente contexto: “Como se sabe el G-8 rechaza la _____ de Antonio Erman González”.

9.2.3 Skip-gram

Por su parte, la arquitectura Skip-gram propone un enfoque inverso con respecto al CBOW. En este caso partimos de un término para predecir los términos que se encuentran en el contexto. Para ello, alimenta la red de neuronas con tuplas del tipo (término de e del modelo. La red calcula las probabilidades con las tuplas de entrada que se le da a cada tupla. Cuantas más ocurrencias de una misma tupla de entrada se den, mayor probabilidad se asignará a la tupla.

Una vez entrenado, el sistema tomará como entrada la representación vectorial de un término, que tendrá la forma de un vector de dimensión N (número de términos totales del dataset).

La salida que generará es un vector de las mismas dimensiones con las probabilidades de ocurrencia del término con cada uno de los términos del corpus. Los pesos de las neuronas de la capa oculta caracterizan la distribución del término y su contexto.

Básicamente predice las palabras de contexto circundantes dentro de una ventana específica dada la palabra actual. (imagen 9.5)

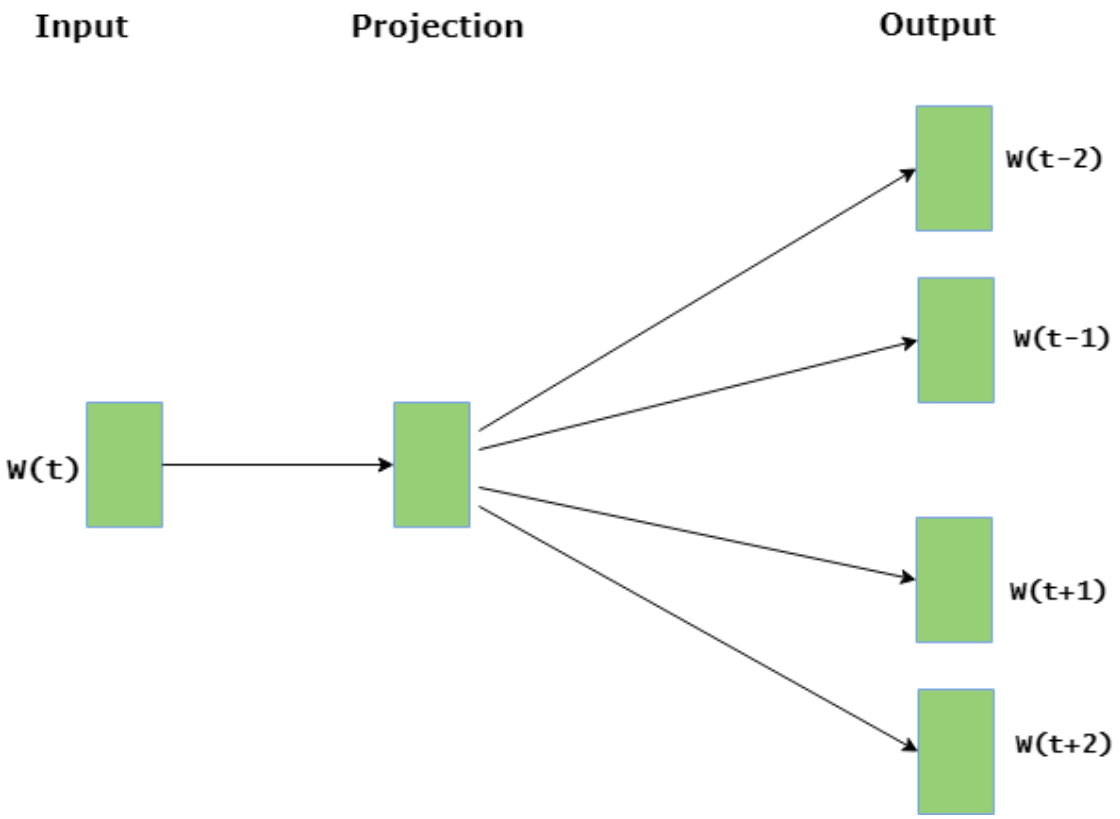


imagen 9.5 arquitectura de Skip-gram.

9.3 Fasttext

Llegado a este punto el lector comprenderá el esfuerzo dedicado a la estructuración de los apartados y siguiendo en detalle cada uno de los capítulos. Aquí en base al data frame recolectado del capítulo 2, nos encargamos de procesar el texto en fasttext. Pero ¿qué es fasttext?:

FastText es una biblioteca gratuita de código abierto de Facebook AI Research para aprender words embeddings y clasificaciones de palabras.

Este modelo permite crear algoritmos de aprendizaje no supervisado o de aprendizaje supervisado para obtener representaciones vectoriales de palabras.

FastText es una extensión de word2vect, es compatible con los modelos CBOW y Skip-gram. Se usa para encontrar similitudes semánticas.

Cada palabra es tratada como la suma de sus composiciones de caracteres llamados ngrams. El vector para una palabra está compuesto por la suma de sus ngrams. Por ejemplo el vector para la palabra “trabajo” está compuesto por la suma los vectores para los ngrams “<tr, tra, trab, trabaj, trabajo>, bajo>, ajo, jo>” (los símbolos < y > representan el inicio y final de ngrams). (imagen 9.6)

$$\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g.$$

Imagen 9.6

De esta forma se espera obtener mejores representaciones para palabras “raras”, las cuales cuentan con muy pocas apariciones en corpus de textos, y así poder generar vectores para palabras que no se encuentran en el vocabulario de los word embeddings. Un ejemplo en español sería: “amistades”, fasttext reconoce el morfema (significado central de la palabra), en este caso el lexema es “amistad” y la asocia a la represantación del vector <amistad>, de esta manera palabras raras dejan de ocupar memoria.

En este trabajo se acudió al data frame generado sin procesar, en particular, anterior a la eliminación de stopwords y lematización. Se usó el texto de los artículos recolectados, a fin de entrenar con fasttext y generar word embeddings por cada palabra en cada artículo.

Algunos de los hiper parámetros utilizados fueron:

- modelo: cbow como arquitectura de entrenamiento.
- tamaño: el tamaño del embedding que se va a aprender (valor: 100)
- Las subpalabras son todas las subcadenas contenidas en una palabra entre el tamaño mínimo (minn) y el tamaño máximo (maxn). Se decidió tomar todas las subpalabras entre 2 y 5 caracteres, pero otro rango podría ser más apropiado para diferentes idiomas:

Luego de obtener unas listas de embeddings (representaciones de palabras por artículos en fasttext), el siguiente paso fue multiplicar la frecuencia de cada palabra retornada en el pre procesamiento LDA en cada tópico (capítulo 5.2) por las words embeddings generados.

Con este topic_vector que representa nuestro nuevo word embedding de cada tópico generamos una suma y dividimos por el tamaño total de artículos en cada tópico, obteniendo un escalar que representa el promedio. Con este promedio por tópico se generaron gráficas que se hablarán en el próximo apartado.

Para ello, se utilizó las siguientes líneas:

Algoritmo 9.1

input: F listas de listas de embeddings procesado por fasttext, C listas de listas de frecuencia de cada topico.
output: vector con la multiplicación entre las dos entradas.

```
for j ∈(0, number_topic) do
    for i ∈(0, number_word):
        a+=(F[j][i]*C[j][i])
    end for
    topics_vector.append(a)
end for
return topics_vector
```

Algoritmo 9.2

input: X: Dimensión de la cantidad de artículos por tópicos.
Y : dimensión de cada tópico.

```
for i∈(1, X) do
    for j ∈(1, Y):
        a += lda_output[i][j]*topics_vector[j]
    end for
endfor
promedio = a/x
return(promedio)
```

Básicamente la ecuación propuesta se observa a continuación (Imagen 9.7).

$$\text{promedio} = \frac{\prod_{k=1}^{t}\prod_{i=1}^{n}\prod_{j=1}^{m} F_{i,j} C_{j,i} O_{i,j}}{n}$$

n = cantidad de tópicos

m= cantidad de palabras

t = cantidad de articulos

F = word embeddings de palabras en fasttext

C = topicos de palabras

O = salida de LDA

imagen 9.7

9.4 Similitud coseno y Distancia coseno

La similitud de coseno usa el coseno del ángulo entre dos vectores en el espacio vectorial como una medida de la diferencia entre los dos individuos. En comparación con la medición de distancia euclidiana, la similitud del coseno presta más atención a la diferencia en la dirección de dos vectores que la distancia o la longitud (imagen 9.7).
El valor del coseno es [-1,1], cuanto más cercano a 1, más pequeño es el ángulo entre los dos vectores y más similar.

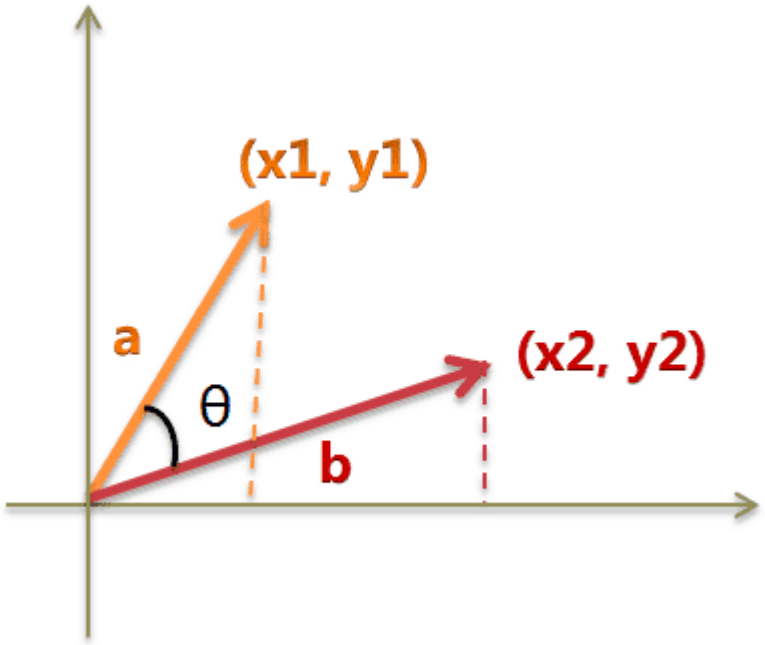


imagen 9.8

La similaridad coseno entre vectores A, B es definida como:

$$\text{similitud\_coseno}\ (A,B) = \frac{\langle A,B \rangle}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Mientras que la distancia coseno es :

$$\text{distancia\_coseno}(A,B) = 1 - \text{similitud\_coseno}(A,B)$$

En minería de textos se aplica la similitud coseno con el objeto de establecer una métrica de semejanza entre textos [23]. En minería de datos se suele emplear como un indicador de cohesión de clústeres de textos. La similitud coseno no debe ser considerada como una métrica debido a que no cumple la desigualdad triangular.

9.5. Distancia euclidiana

La distancia euclidiana es la distancia entre dos o más puntos en el plano.

Sean puntos $A(x_1, y_1)$ con $B(x_2, y_2)$. Entonces, la distancia euclidiana entre A y B es $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Si son tres puntos en el espacio tridimensional $A(x_1, y_1, z_1)$ con $B(x_2, y_2, z_2)$. Entonces, la distancia euclidiana entre A y B es $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ (imagen 9.8); Si se extiende a un espacio de alta dimensión, la fórmula se puede deducir por analogía,

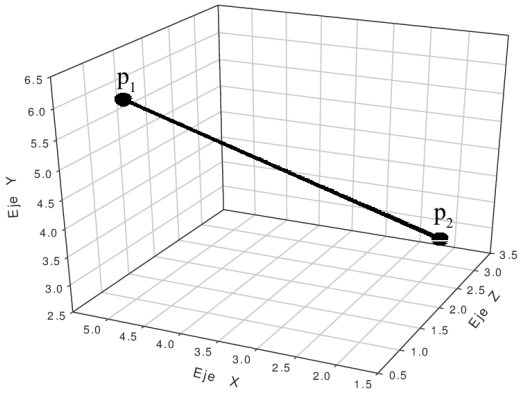


imagen 9.9
[28]

9.6. Usos prácticos de similaridad y distancia.

Se utilizó tanto la similitud coseno como la euclidiana para evaluar puntos en el espacio bidimensional.

Para ello, obteniendo los resultados del algoritmo 9.2 de todos los periodos, donde se representa el promedio en cada tópico, logramos comparar la similitud coseno (y distancia euclidiana) sobre dos periodos de la misma época pero distintos diarios, ya que estos representan un punto en el espacio.

10 Top2vec otro modelo de extracción de tópicos.

Si bien pudimos obtener resultados en el modelo de LDA (que es un modelo probabilístico) no fueron satisfactorios a la hora de analizar con el experto de dominio. No se encontraron patrones de interés en dicho análisis de cada tópico.

La capacidad de organizar, buscar y resumir un gran volumen de texto es un verdadero problema incluso para LDA.[24]

Top2vec, aprovecha la incrustación semántica conjunta de documentos y embeddings para encontrar vectores temáticos. El modelo encuentra automáticamente la cantidad de temas como veremos más adelante. Los vectores de temas resultantes se incrustan conjuntamente con los vectores de documentos y palabras, con una distancia entre ellos que representa la similitud semántica. Top2vec muestra que sus resultados son significativamente más informativos y representativos del corpus entrenado que los modelos probabilísticos.

10.1 Top2vec vs LDA

El método de modelado de temas más utilizado es la asignación de Dirichlet latente (LDA) [25]. Es un modelo probabilístico generativo que describe cada documento como una mezcla de temas y cada tema como una distribución de palabras.

Estos modelos asumen el número de temas a conocer. La discretización de los temas es necesaria para modelar la relación entre documentos y palabras. Esta es una de las mayores debilidades de estos modelos, ya que rara vez se conoce el número de temas o la forma de estimarlo, especialmente para conjuntos de datos muy grandes o desconocidos puede traer muchos problemas como abordamos más arriba.

Cada tema producido por estos métodos es una distribución de probabilidades de palabras. Como tal, las palabras de mayor probabilidad en un tema suelen ser palabras como economía, sociedad y otras palabras comunes en el idioma [26]. Estas palabras comunes, también llamadas palabras vacías, a menudo necesitan filtrarse para hacer que los temas sean interpretables y extraer las palabras informativas del tema. Encontrar el conjunto de palabras vacías que deben eliminarse no es un problema trivial, ya que es específico tanto del lenguaje como del corpus [27]; un modelo de tema entrenado en texto sobre presidentes probablemente tratará a Cristina como una palabra vacía, ya que no es muy informativo. LDA y PLSA utilizan representaciones de bolsa de palabras (BOW) de documentos como entrada que ignoran la semántica de palabras. En la representación BOW, las palabras argentina y argentinas se tratarán como palabras diferentes, a pesar de su similitud semántica. Las técnicas de derivación y

lematización tienen como objetivo abordar este problema, pero a menudo hacen que los temas sean más difíciles de entender. Además, la lematización no reconoce la similitud de palabras como banco (unidad financiera) y banco (de alguna plaza), que no comparten una raíz de palabra.
El objetivo de los modelos generativos probabilísticos como LDA y PLSA es encontrar temas que puedan usarse para recrear las distribuciones de palabras del documento original con un error mínimo. Sin embargo, una gran proporción de todo el texto contiene palabras poco informativas que pueden no considerarse de actualidad.
Estos modelos no diferencian entre palabras informativas y no informativas, ya que su objetivo es simplemente recrear las distribuciones de palabras del documento. Por lo tanto, nos encontramos al probar el modelo LDA que las palabras de alta probabilidad en los temas que se encontraban no corresponden necesariamente a lo que el experto de dominio pensaría intuitivamente como tópico.

10.2 ¿Cómo funciona Top2Vec?

Top2Vec es un algoritmo que detecta temas presentes en el texto y genera vectores de tema, documento y word embeddings conjuntamente.

El algoritmo realiza los siguientes pasos para descubrir temas en una lista de documentos:

- Genera embedding vectors para documentos y palabras.
- Realiza la reducción de dimensionalidad en los vectores utilizando el algoritmo UMAP.
- Agrupa los vectores usando un algoritmo de agrupamiento HDBSCAN.
- Asigna temas a cada grupo.

Veamos en detalle estos puntos.

10.2.1 Generar embeddings vectors para documentos y palabras.

Un embeddings vectors es un vector que nos permite representar una palabra o un documento de texto en un espacio multidimensional. La idea detrás del embedding vectors es que palabras o documentos de texto similares tendrán vectores similares. (figura 10.1)

La creación de embeddings vectors para cada documento nos permite tratar cada documento como un punto en un espacio multidimensional. Top2Vec también crea vectores de embeddings vectors conjuntamente, lo que nos permite determinar las palabras clave del tema más adelante. (Imagen 10.2)

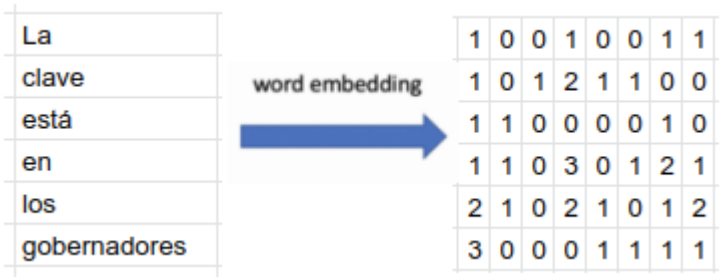


Imagen 10.1

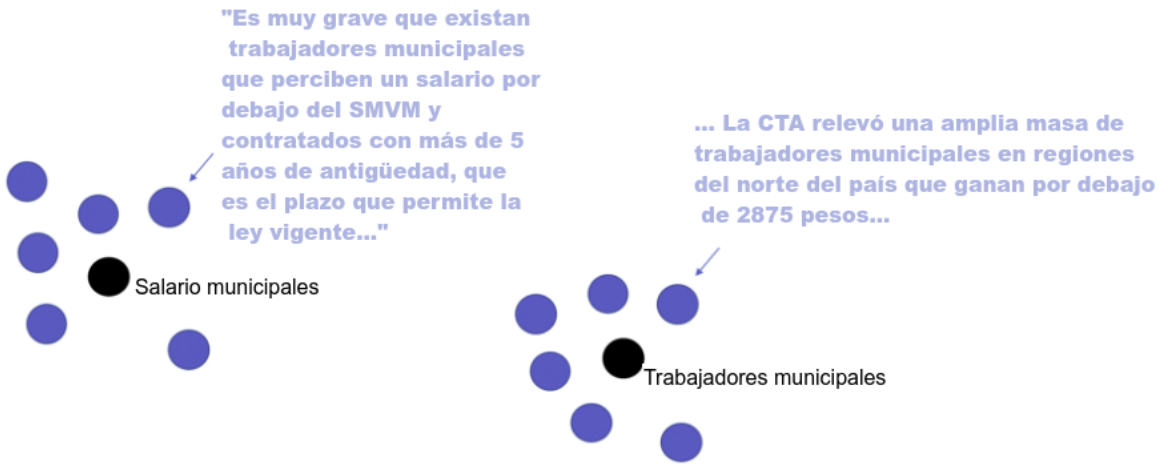


Imagen 10.2

Una vez que tenemos un conjunto de vectores de palabras y documentos, podemos pasar al siguiente paso.

10.2.3 Realizar reducción de dimensionalidad

Una vez que tengamos vectores para cada documento, el siguiente paso natural sería dividirlos en grupos mediante un algoritmo de agrupación. Sin embargo, los vectores generados desde el primer paso pueden tener hasta 512 componentes según el modelo de embeddings que se utilizó.

Por esta razón, tiene sentido realizar algún tipo de algoritmo de reducción de dimensionalidad para reducir el número de dimensiones en los datos. Top2Vec utiliza un algoritmo llamado UMAP (Aproximación y proyección de variedad

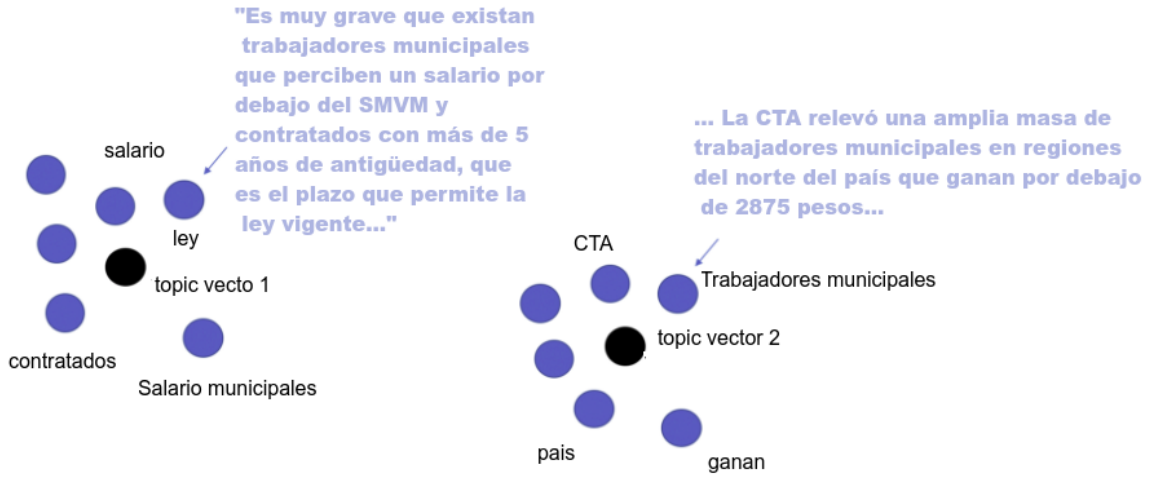
uniforme) para generar vectores de incrustación de menor dimensión para cada documento.

10.2.4 Clusters de vectores

Top2Vec usa HDBSCAN, un algoritmo de agrupamiento jerárquico basado en la densidad, para encontrar áreas densas de documentos. HDBSCAN es básicamente una extensión del algoritmo DBSCAN que lo convierte en un algoritmo de agrupamiento jerárquico. El uso de HDBSCAN para el modelado de temas tiene sentido porque los temas más grandes pueden constar de varios subtemas.

10.3.5 Asignación de temas a cada cluster

Una vez que tengamos grupos para cada documento, podemos simplemente tratar cada grupo de documentos como un tema separado en el modelo de tema. Cada tema se puede representar como un vector de tema que es esencialmente sólo el centroide (punto promedio) de los documentos originales que pertenecen a ese grupo de temas. Para etiquetar el tema usando un conjunto de palabras clave, podemos calcular las n palabras más cercanas al vector centroide del tema.



Topic vector 1: Salario_municipales, ley, salario, contratados.

Topic vector 2: ganan, trabajadores_municipales, pais, CTA

Si bien Top2Vec es mucho más complejo que el enfoque LDA estándar para el modelado de temas, puede brindarnos mejores resultados, ya que los vectores incrustados para palabras y documentos pueden capturar de manera efectiva el significado de palabras y frases.

10.4 Encontrar el número de tópicos

La semantic embedding tiene la ventaja de aprender una representación continua de temas. En el espacio vectorial de documento y palabra integrado conjuntamente, los documentos y las palabras se representan como posiciones en el espacio semántico. En este espacio, cada vector de documento puede verse como una representación del tema del documento. Los vectores de palabras más cercanos a un vector de documento son los que mejor describen semánticamente el tema del documento.

En el espacio semántico, un área densa de documentos puede interpretarse como un área de documentos muy similares. Esta área densa de documentos indica un tema que es común a los documentos. Dado que los vectores de documentos representan los temas de los documentos, se puede calcular el centroide o promedio de esos vectores. Este centroide es el vector de tema que es más representativo del área densa de documentos a partir de los cuales se calculó. Las palabras más cercanas a este vector de tema son las palabras que mejor lo describen semánticamente. La suposición principal detrás de top2vec es que la cantidad de áreas densas de vectores de documentos es igual a la cantidad de temas destacados. Esta es una forma natural de discretizar temas, ya que se encuentra un tema para cada grupo de documentos que comparten un tema destacado. Para encontrar las áreas densas de los documentos en el espacio semántico, se utiliza el agrupamiento basado en la densidad en los vectores del documento, (HDBSCAN)

10.5 Guardando de resultados de top2vec.

Una de las facetas importantes en la que concluyó esta tesis, es mostrar resultados. De igual manera que en el apartado 11 , se utilizaron los resultados guardados del dataframe de links pre procesados y usamos el lematizador, además de quitar las palabras vacías.
Todo ello con el objetivo, al igual, que para LDA, poder entrenar con el modelo top2vec y obtener resultados. Con el fin de mostrar a nuestro experto de dominio, se logré guardar los resultados de cada tópico por periodo en un archivo formato imágen, a su vez en un archivo csv se guardó cada periodo de todos los periodos obtenidos en el tópico.
Se realizó un conteo particular, para obtener los conteos de cada periodo de cada diario, por tópicos, resultando otro archivo csv.

11. Estructura de la Tesis

En primer lugar abordaremos la construcción del dataset y los desafíos que implicó construirlo desde cero.

En el capítulo 2 analizaremos el proceso de construcción para obtener los artículos. Este capítulo encierra el procedimiento para extraer información de los diarios a estudiar, como así errores que se obtuvieron y su solución. En el capítulo 3 tenemos el preprocesamiento del texto ya escrapeado y almacenado. Este capítulo nombra distintas herramientas que se usaron en el tratamiento de texto como palabras vacías, tokenización, lematización, stem. Esto abre camino al siguiente capítulo que trata de los bigramas y collocation. En este último nombraremos además otros métodos de collocation.

En el capítulo 5 nos adentramos de lleno al algoritmo de aprendizaje automático. Nos encontramos forzados a incluir, en este nivel, un marco teórico e histórico para inmediatamente después poder describir las matemáticas detrás de LDA como así su estructura e hiper parámetros.

Usamos el algoritmo de LDA para obtener tópicos. Para sacar la cantidad óptima de tópicos, se desarrollará este tema en la sección 6.

En el capítulo 7 , nos introducimos en Sklearn, la librería usada en este trabajo, como así también en este apartado nombraremos las variables seteadas como el número de tópicos, alfa, beta.

Mientras que en el capítulo 8, mostramos el marco práctico de las herramientas utilizadas para la visualización.

En el capítulo 9 , nos adentramos a FastText y toda su teoría para abordar el tema de words embeddings con ejemplos básicos, introducimos los modelos del lenguaje que trabajamos. Finalizando este capítulo con una explicación de distancias euclidianas y distancia coseno, además del armado del topic_vector con un pseudocódigo.

Mientras que en el capítulo 10 nos topamos con definiciones de top2vec , otro modelo para la extracción de tópicos. Asumimos una breve comparación entre LDA vs top2vec , su funcionamiento como así explicamos su búsqueda automática de el número de tópicos.

Por otro lado, en el capítulo 11, mostraremos cómo fueron guardados los resultados, para cada paso que hicimos.

Por último se muestra una serie de experimentos , en las cuales se usaron librerías como gensim, sklearn y top2vec.

11.1 Guardando resultados en un archivo.

Un primer paso fue la extracción de los links. Cada link tiene por detrás el enlace a algún artículo de diarios que con la librería selenium webdriver se extrajo. Los enlaces obtenidos se guardaron en un dataframe, para posteriormente generar un archivo csv el cuál contenía esta data según la búsqueda.

La segunda parte, al extraer el archivo csv de artículos de los links, se obtuvo un dataframe base donde cada columna representa atributos pertenecientes a la investigación. Este data frame generado se guardó en un archivo .csv para posteriormente analizar su contenido. En esta parte, para el scrapeo se utilizó la librería de python: “newspaper 3k” (capítulo 2.3)

En el csv, una columna llamada “Text” contiene el cuerpo del artículo, esta variable será útil para un previo análisis. Tener un registro de las palabras ya enlistadas para un previo análisis tiene sus ventajas. En dicha lista se hizo un conteo de ocurrencias con la librería Counter de python y se guardó en distintos archivos .txt, según la época y niveles.

En una cuarta parte, utilizamos al libreria pyLDAvis para obtener una visualización altamente interactiva de los clústeres creados por LDA. Esta visualización se guardó en un archivo .html para luego ser analizado.

En la quinta etapa se efectuó un desarrollo con la librería de WordCloud de Python. Dicha investigación mantiene una nube de palabras por cada tópico, por cada periodo. En esta etapa también se guardó también la frecuencia de palabras denotadas por el procesamiento de esta librería.

En la sexta etapa y última extracción de archivos, utilizamos la librería fasttext, está librería nos permitió no solo obtener las words embeddings de palabras por artículo como se explicó en el apartado anterior, además, nos permitió guardar el modelo entrenado para su posterior análisis. En esta sección se crearon imágenes donde dan sentido a los word embeddings con Fasttext. (véase capítulo 9)

12. Resultados experimentales

A continuación se mostrará la construcción del modelo. A saber, se usaron 3 modelos. En dos de estos modelos se desarrolló con la librería sklearn, mientras que para el otro se utilizó la librería gensim (véase capítulo 7). Ambas librerías trabajan de manera independiente y son formalmente distintas.
Para lograr el objetivo de este trabajo, tuvimos que pre procesar cada artículo, para poder pasar como inputs a los modelos. En los siguientes ítems nombraremos partes específicas donde se obtuvieron casos experimentales desde un dataframe al entrenamiento.

12.1. Estructura de separación de artículos.

Luego de obtener los links con la herramienta de selenium, nuestro siguiente paso es poder dividir por periodos, para eso se determinaron los años acorde al experto de dominio. Además de que él mismo nos proporcionó niveles de importancias sobre cada links analizado previamente. Tales niveles sugieren la importancia de los artículos:

1 muy importante

2 importante

3 levemente importante

4 menos importante

Aquí la utilidad de trabajar en un dataframe, fue de uso exponencial. La librería pandas proporciona un filtrado de dichos valores.

Se filtraron los enlaces de nivel 1 y 2 en el grupo 1, mientras que en el grupo 2 se filtraron los enlaces de nivel 1, 2 y 3. Olvidándonos definitivamente del nivel 4.

Obtenido estos grupos se calculó las cantidades de cada nivel, siendo estas cantidades como se muestran en la imagen siguiente:

```
pagina 12
cantidad de pagina por nivel
level 1 :  743
level 2 :  712
level 3 :  389
level 4 :  716
suma nivel 1 y 2:  1455
suma nivel 1 y 2 y 3:  1844
suma nivel 1 y 2 y 3 y 4:  2560
```

```
 LN
cantidad de pagina por nivel
level 1 :  2873
level 2 :  348
level 3 :  233
level 4 :  205
suma nivel 1 y 2:  3221
suma nivel 1 y 2 y 3:  3454
suma nivel 1 y 2 y 3 y 4:  3659
```

Obtenida la separación por nivel, se agrupó por periodos, como mostramos en la siguiente imagen.

Página: Pagina12

```
grupo 1
periodo 1 (1995-1999):  13
periodo 2 (2000-2002):  24
periodo 3 (2003-2007):  130
periodo 4 (2008-2011):  505
periodo 5 (2012-2015):  582
periodo 6 (2015-2019):  194
periodo 7 (2019-2020):  6

Grupo 2
periodo 1 (1995-1999):  13
periodo 2 (2000-2002):  24
periodo 3 (2003-2007):  233
periodo 4 (2008-2011):  605
periodo 5 (2012-2015):  723
periodo 6 (2015-2019):  238
periodo 7 (2019-2020):  6
```

Página: La Nación

```
grupo 1
periodo 1 (1995-1999):  748
periodo 2 (2000-2002):  399
periodo 3 (2003-2007):  302
periodo 4 (2008-2011):  130
periodo 5 (2012-2015):  155
periodo 6 (2015-2019):  1161
periodo 7 (2019-2020):  326

Grupo 2
periodo 1 (1995-1999):  774
periodo 2 (2000-2002):  412
periodo 3 (2003-2007):  333
periodo 4 (2008-2011):  152
periodo 5 (2012-2015):  185
periodo 6 (2015-2019):  1232
periodo 7 (2019-2020):  366
```

imagen 12.1

No obstante se decidió avanzar sobre el grupo 1, que abarca los niveles 1 y 2 de nivel de importancia en artículos clasificados por el experto de dominio. Calculando un promedio de artículos lo que se observa en la siguiente imagen:

```
grupo 1 Pagina12
periodo 1 (1995-1999):  13
periodo 2 (2000-2002):  24
periodo 3 (2003-2007):  130
periodo 4 (2008-2011):  505
periodo 5 (2012-2015):  582
periodo 6 (2015-2019):  194
periodo 7 (2019-2020):  6

promedio de articulos (1995-2021):  207.71428571428572

Grupo 2 LaNacion
periodo 1 (1995-1999):  748
periodo 2 (2000-2002):  399
periodo 3 (2003-2007):  302
periodo 4 (2008-2011):  130
periodo 5 (2012-2015):  155
periodo 6 (2015-2019):  1161
periodo 7 (2019-2020):  326

promedio de articulos (1995-2021):  460.14285714285717
```

Imagen 12.2

12.2 Primeros pasos

Luego de obtener un dataframe, donde en cada columna representa el preprocesamiento hablado en el capítulo 3. Esto es, Tokenización, sacar palabras vacías, puntuación, normalizar el texto, Lematización o stemming, bigramas. Estos mismos se desarrollan en el capítulo 3.
Tuvimos que desarrollar stemming debido a que en la lematización aparecían palabras raras como Cristian Fernandez. Sin embargo, nos concentramos más en dichas palabras debido a que con stemming perdíamos el significado de la palabra en sí, la misma traería ciertas dificultades al procesar por un modelo.

	Title	Text	Salio_de	Dates	Hours_publish	year	text_processed	lemmatiz	tokens_punct	text_sin_acentos	name	remove_stop_words	stemmed
0	Página	Por Maximiliano Montenegro Como nunca antes ...	pagina12	1998-04-07	00:00:00	1998.0	Por Maximiliano Montenegro Como nunca antes ...	Por Maximiliano Montenegro Como nunca antes...	por maximiliano montenegro como nunca antes du...	por maximiliano montenegro como nunca antes du...	por maximiliano montenegro como nunca antes du...	maximiliano montenegro convertibilidad fondo m...	[maximilian, montenegr, convertibil, fond, mon...
1	Página	Por Maximiliano Montenegro Desde Washington Ro...	pagina12	1998-04-17	00:00:00	1998.0	Por Maximiliano Montenegro Desde Washington Ro...	por Maximiliano Montenegro Desde Washington Ro...	por maximiliano montenegro desde washington ro...	por maximiliano montenegro desde washington ro...	por maximiliano montenegro desde washington ro...	maximiliano montenegro washington roque fernan...	[maximilian, montenegr, washington, roqu, fern...
2	Suplemento Económico de Página/12	Roggio contra la reforma de Erman: "NO SIRVE P...	pagina12	1998-05-17	00:00:00	1998.0	Roggio contra la reforma de Erman: "NO SIRVE P...	Roggio contra el reforma de Erman : " no sirve...	roggio contra el reforma de erman no sirve par...	roggio contra el reforma de erman no sirve par...	roggio contra el reforma de erman no sirve par...	roggio reforma erman sirve nada raul dellator...	[roggi, reform, erman, sirv, nada, raul, dell...
3	Página	Por David Cufré Resultó premonitoria la elec...	pagina12	1998-07-23	00:00:00	1998.0	Por David Cufré Resultó premonitoria la elec...	Por David Cufré resultar premonitorio el el...	por david cufré resultar premonitorio el elecc...	por david cufre resultar premonitorio el elecc...	por david cufre resultar premonitorio el elecc...	david cufre resultar premonitorio eleccion cir...	[dav, cufr, result, premonitori, eleccion, cir...
4	Página	No hubo cruces fuertes ni críticas alzando l...	pagina12	1998-07-25	00:00:00	1998.0	No hubo cruces fuertes ni críticas alzando l...	no haber cruz fuerte ni crítica alzar el vo...	no haber cruz fuerte ni crítica alzar el voz c...	no haber cruz fuerte ni critica alzar el voz c...	no haber cruz fuerte ni critica alzar el voz c...	cruz fuerte critica alzar voz reunion tenso co...	[cruz, fuert, critic, alzar, voz, reunion, ten...
5	Página	Por Fernando Almirón Todo indica que hoy,...	pagina12	1998-08-26	00:00:00	1998.0	Por Fernando Almirón Todo indica que ho p...	Por Fernando Almirón todo indicar que h...	por fernando almirón todo indicar que ho por t...	por fernando almiron todo indicar que ho por t...	por fernando almiron todo indicar que ho por t...	fernando almiron indicar tercero consecutivo l...	[fern, almiron, indic, tercer, consecut, ley, ...
6	Buenos Aires	Refotms menem	pagina12	1998-08-27	00:00:00	1998.0	Refotms menem	Refotms menem	refotms menem	refotms menem	refotms menem	refotms menem	[refotms, menem]
7	Página	Por Fernando Almirón Ayer naufragó por te...	pagina12	1998-08-27	00:00:00	1998.0	Por Fernando Almirón Ayer naufragó por te...	Por Fernando Almirón ayer naufragar por...	por fernando almirón ayer naufragar por tercer...	por fernando almiron ayer naufragar por tercer...	por fernando almiron ayer naufragar por tercer...	fernando almiron naufragar tercero consecutivo...	[fern, almiron, naufrag, tercer, consecut, tra...
8	Página	Por Maximiliano Montenegro No resuelve el de...	pagina12	1998-08-31	00:00:00	1998.0	Por Maximiliano Montenegro No resuelve el de...	por Maximiliano Montenegro no resolver el d...	por maximiliano montenegro no resolver el des...	por maximiliano montenegro no resolver el des...	por maximiliano montenegro no resolver el des...	maximiliano montenegro resolver desempleo la...	[maximilian, montenegr, resolv, desempleo, l...
9	Página	Por Fernando Almirón Después de tres post...	pagina12	1998-09-03	00:00:00	1998.0	Por Fernando Almirón Después de tres post...	Por Fernando Almirón después de tres po...	por fernando almirón después de tres postergac...	por fernando almiron despues de tres postergac...	por fernando almiron despues de tres postergac...	fernando almiron postergacion consecutivo cuar...	[fern, almiron, postergacion, consecut, cuart,...
10	Página	Por Fernando Almirón El Poder Ejecutivo v...	pagina12	1998-09-23	00:00:00	1998.0	Por Fernando Almirón El Poder Ejecutivo v...	Por Fernando Almirón el Poder Ejecutivo...	por fernando almirón el poder ejecutivo vetar ...	por fernando almiron el poder ejecutivo vetar ...	por fernando almiron el poder ejecutivo vetar ...	fernando almiron ejecutivo vetar articulo reci...	[fern, almiron, ejecut, vet, articul, recient,...

imagen 12.3

Entonces, la columna de interés es la nombrada: “remove_stop_words”, en ella encontramos listas como la siguiente:

['encuesta','sindicato','carlos','tomado','acabar','encuesta','surgir','claramente','trabajador','asalariado','continuar','privilegiar','papel','sindicato,'nacional','acompañado','instancia','gremial','negociacion_colectivo','opinion','expresar','gobierno','enfrascado','reforma','tendiente','debilitar','papel','gremio','nivel','nacional','negociacion_convenio', 'cualquiera', 'suponer', 'discurso', 'oficial', 'negociacion_colectivo', 'permanecer', 'año', 'cerradamente', 'centralizado', 'caracter', 'nacional', 'oficial', 'ministerio', 'año', 'negociacion', 'nivel_empresa',..] ,

que representan de un artículo, “palabras más representativas” del artículo, sin stopwords, puntuaciones, ni números. A continuación, procesamos en un modelo para obtener tópicos.

12.3 Entrenando LDA con sklearn , gensim, top2vec

Una de las complejidades de este trabajo, fue encontrar un modelo que se ajuste a nuestras necesidades, como así el número óptimo de parámetros y de tópicos.

Luego de setear con parámetros adecuados, se obtuvo con el modelo de gensim un resultado que no fue el deseado, debido que no se encontraba ninguna concordancia entre los periodos. A continuación mostramos dichos resultados:

12.3.1 Resultados de pyLDAvis.

En este apartados veremos cómo usar LDA de gensim y pyLDAvis se unen para crear visualizaciones de clústeres de modelado de temas. (véase capítulo 8)

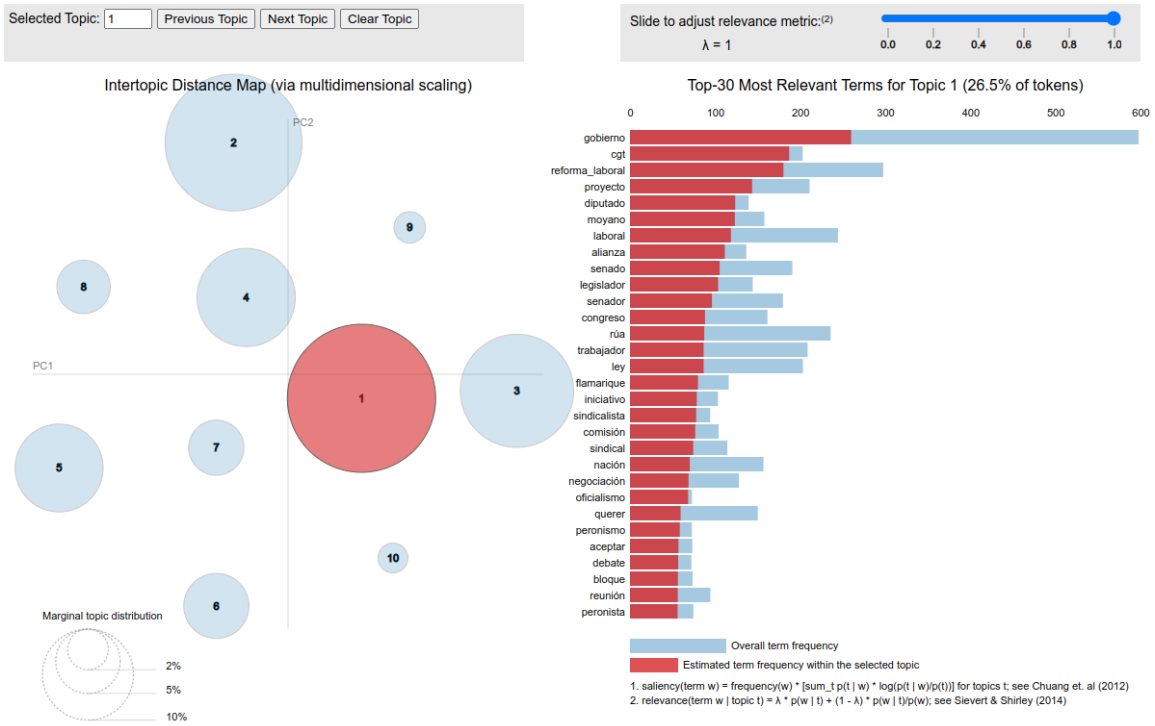


Imagen 12.4 Diario la Nación, tópico 1, período 2000-2002. A la derecha se muestra la frecuencia de palabras.

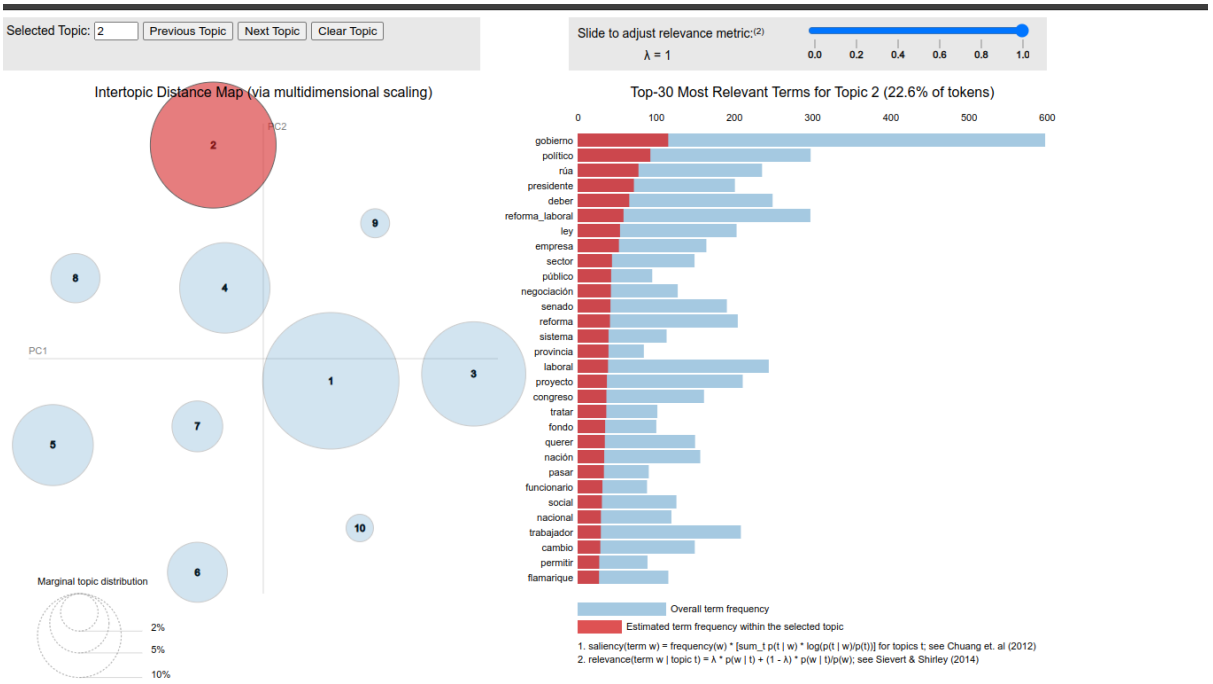


Imagen 12.5 Diario la Nación, tópico 2, período 2000-2002. A la derecha se muestra la frecuencia de palabras.

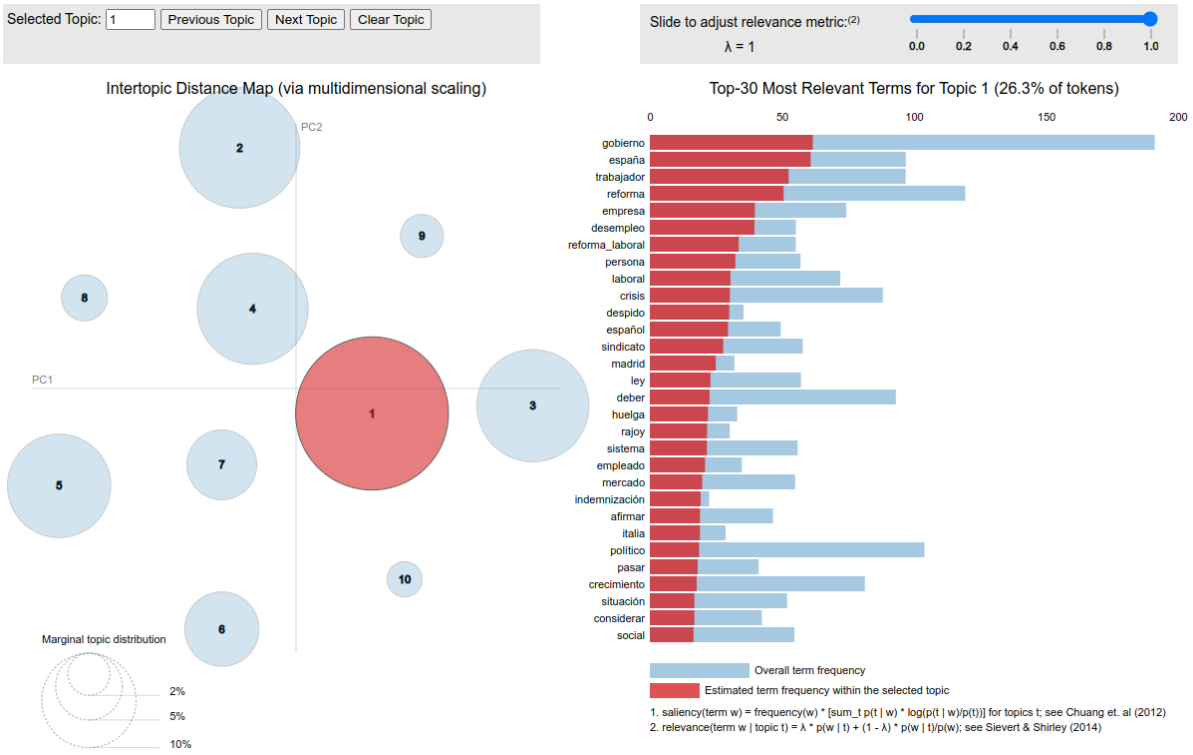


Imagen 12.6 Diario la Nación, tópico 1, período 2012-2016. A la derecha se muestra la frecuencia de palabras.

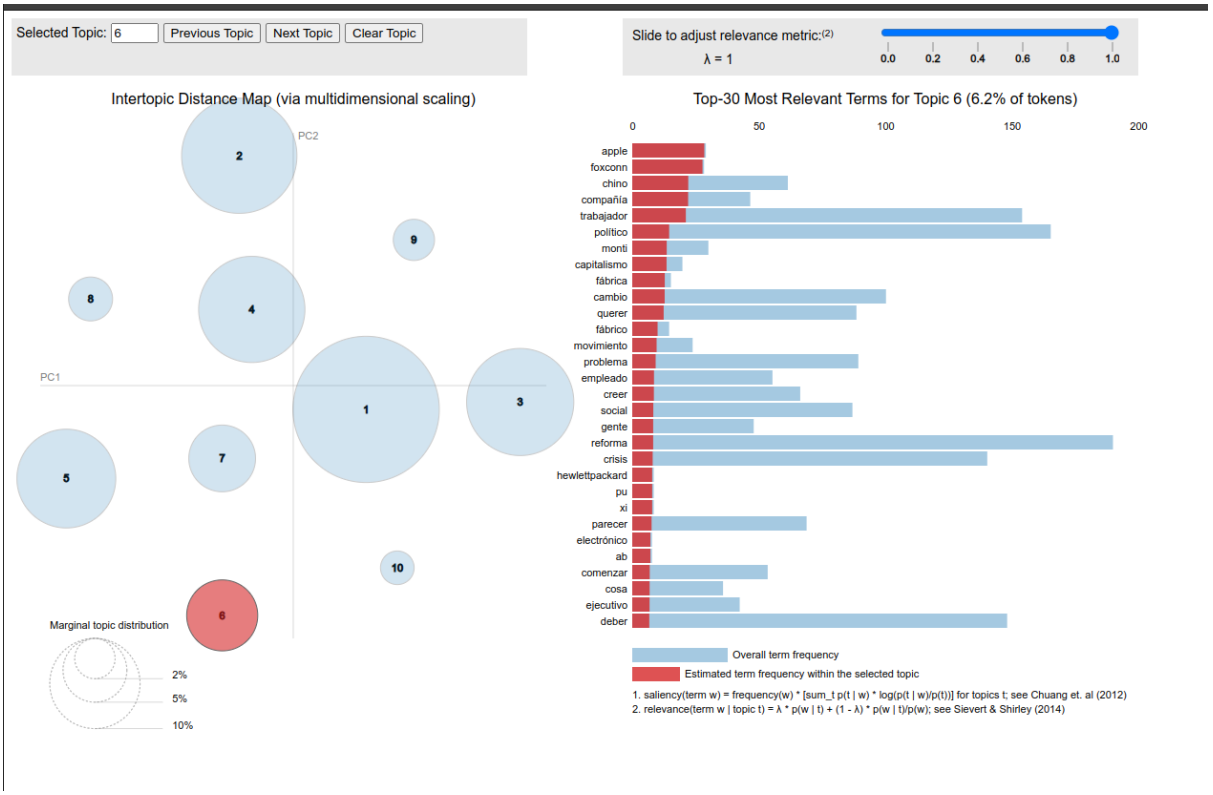


Imagen 12.4 Diario la Nación, tópico 4, período 2012-2016. A la derecha se muestra la frecuencia de palabras.

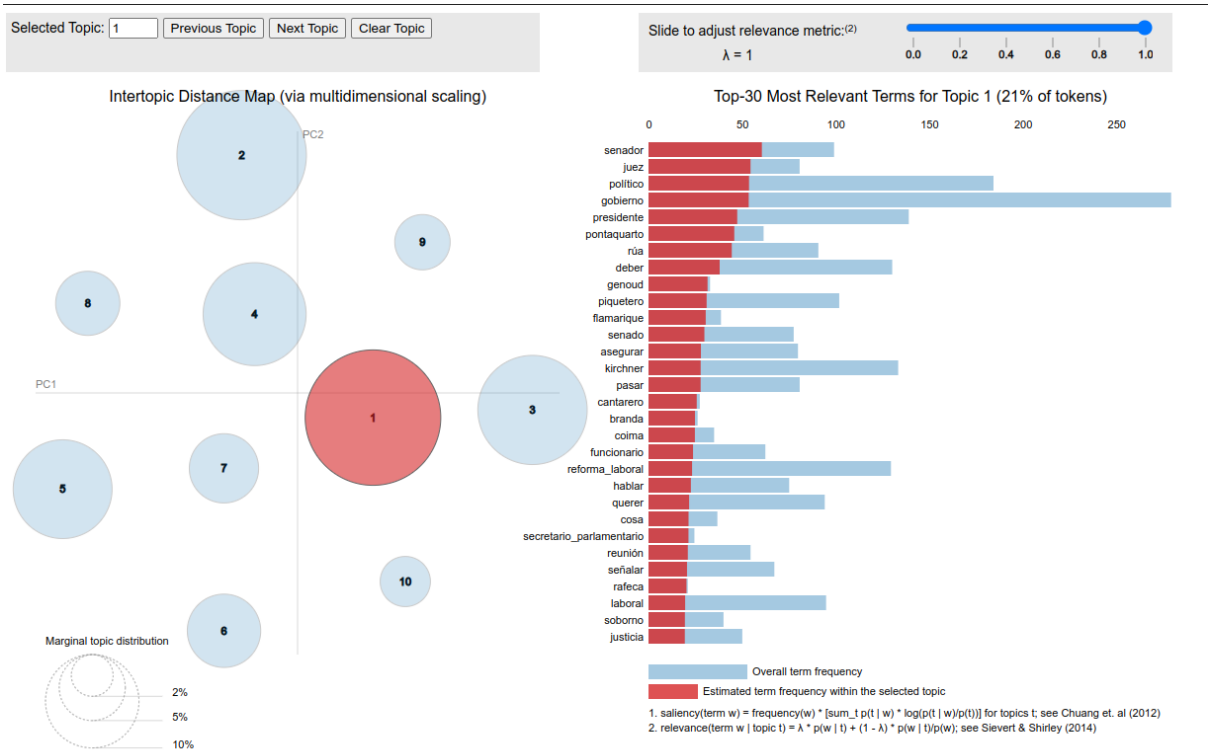


Imagen 12.5 Diario página 12, tópico 1, período 2003-2008. A la derecha se muestra la frecuencia de palabras.

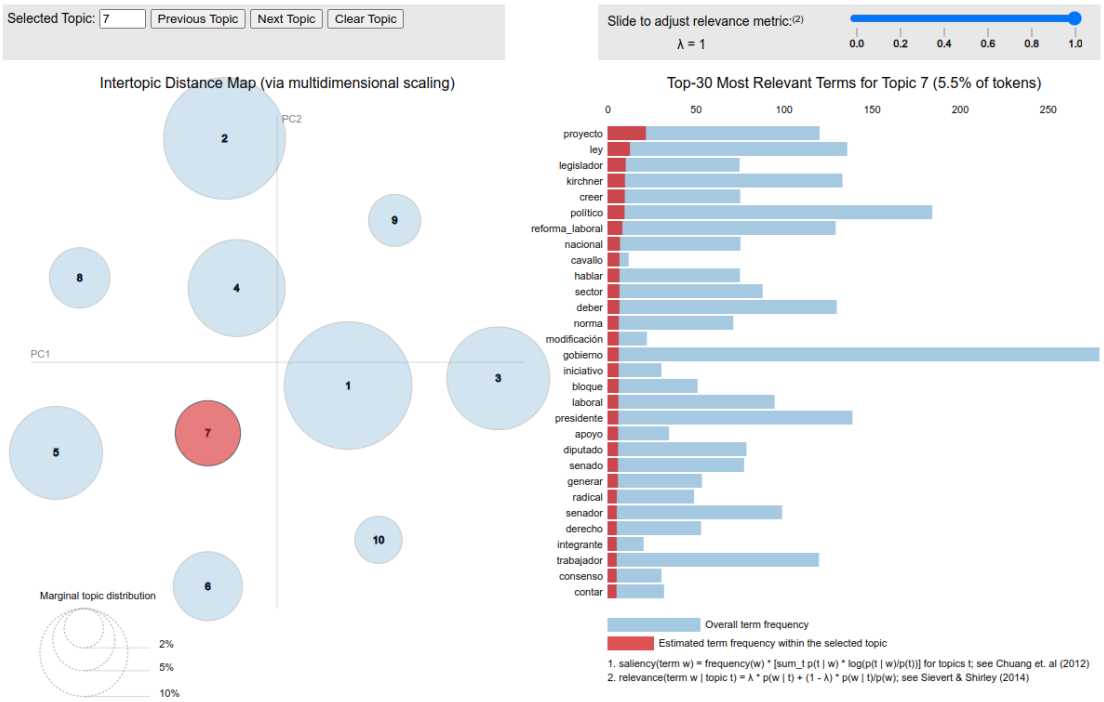


Imagen 12.6 Diario página 12, tópico 7, período 2003-2008. A la derecha se muestra la frecuencia de palabras.

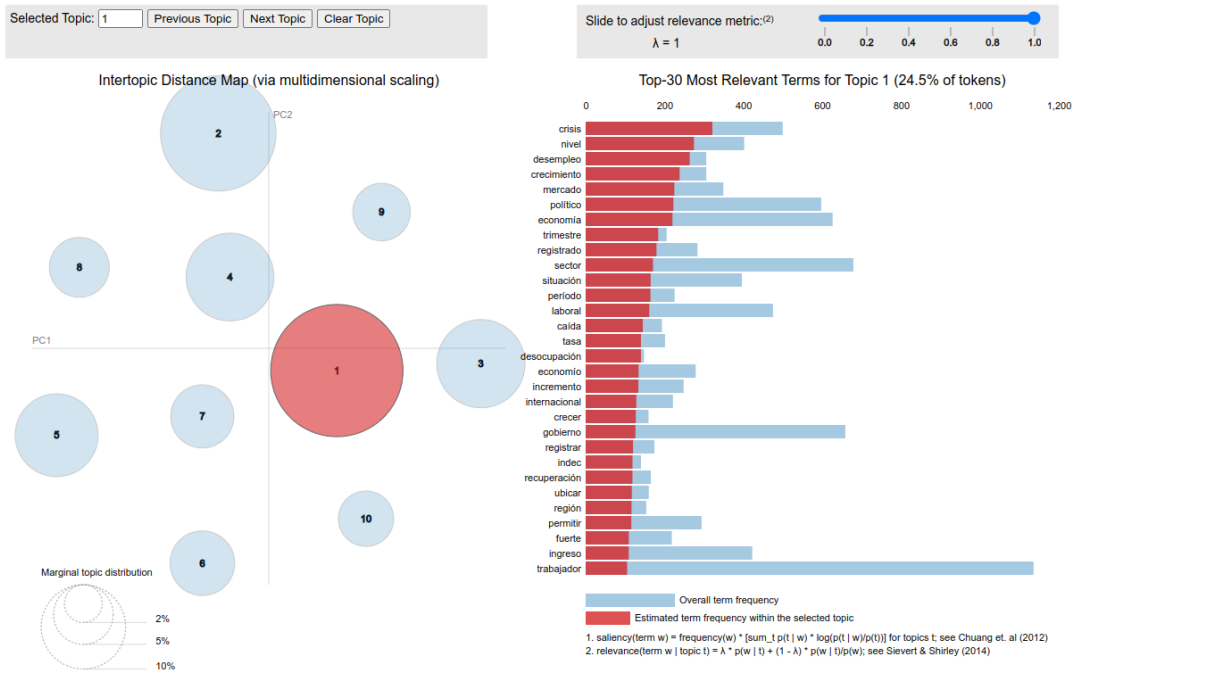


Imagen 12.7 Diario página 12, tópico 1, período 2008-2012. A la derecha se muestra la frecuencia de palabras.

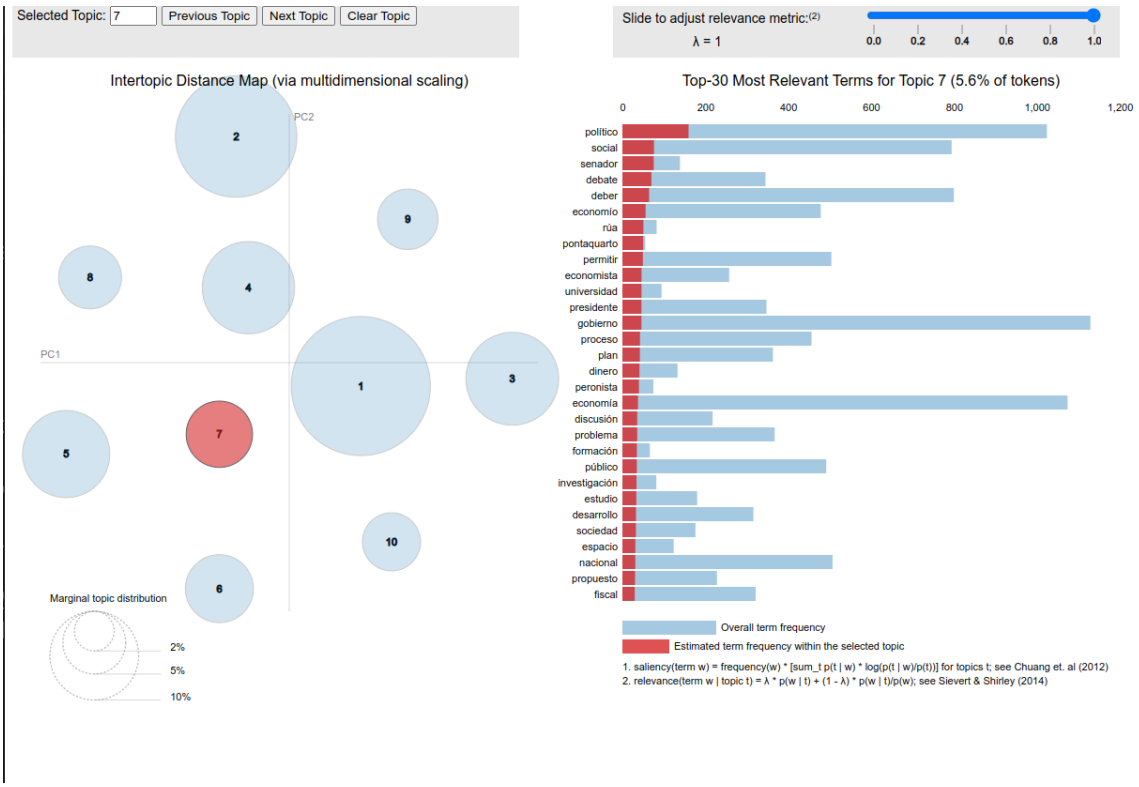


Imagen 12.8 Diario página 12, tópico 7, período 2008-2012. A la derecha se muestra la frecuencia de palabras.

12.3.2 Representación de la información en tabla y wordcloud.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	gobierno	cgt	reforma_laboral	proyecto	diputado	moyano	laboral	alianza	senado	legislador
Topic 2	rúa	presidente	clinton	europa	combustible	alvarez	ministro	villaverde	político	jorge
Topic 3	senador	soborno	senado	juez	cafiero	coima	escándalo	liporaci	sospechar	pedir
Topic 4	gobierno	político	rúa	presidente	deber	reforma_laboral	ley	empresa	sector	público
Topic 5	crecimiento	gobierno	mercado	deber	fiscal	reforma	economía	economío	crecer	creer
Topic 6	gobierno	político	social	creer	presidente	deber	situación	gente	partido	frepaso
Topic 7	trabajador	gobierno	laboral	deber	político	reforma	rúa	sistema	empresa	desempleo
Topic 8	clinton	comercio	rúa	trabajador	mundial	spot	mercosur	préstamo	bloque	publicidad
Topic 9	moyano	mercado	empresa	deber	gobierno	brasil	compañía	negocio	problema	director
Topic 10	político	gobierno	reforma	sindicato	trabajador	italia	berlusconi	ministro	italiano	izquierdo

imagen 12.9 lista de frecuencias , dada por la nube de palabras del diario de La Nación dado en el periodo 2000-2002. Estas listas las usaremos más adelante.

En la imagen 12.10 se observa el topic 1. De acuerdo a la imagen anterior Gobierno subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

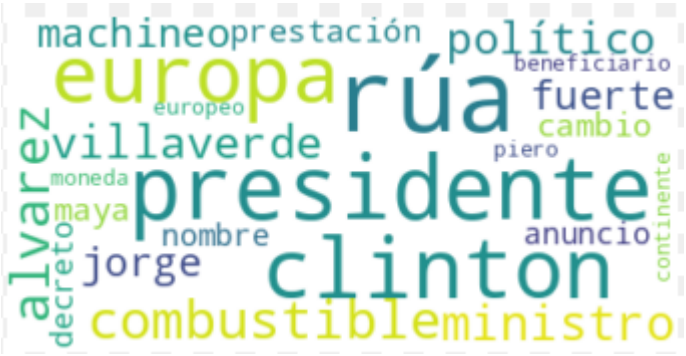


Imagen 12.11 se observa el topic 2. De acuerdo a la imagen 12.9 la palabra rúa preside sobre las presidente, clinton, europa.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	laboral	comercial	secundario	escuela	economía	bachillerato	brasil	cambio	nivel	código
Topic 2	grecia	griego	europeo	euro	zona_euro	deuda	acreedor	gobierno	reforma	bce
Topic 3	gobierno	españa	trabajador	reforma	empresa	desempleo	reforma_laboral	persona	laboral	crisis
Topic 4	españa	crisis	gobierno	europeo	español	grecia	rescate	banco	deber	ue
Topic 5	apple	foxconn	chino	compañía	trabajador	político	monti	capitalismo	fábrica	cambio
Topic 6	indio	modi	chino	político	libertad_económ	corrupción	india	partido	ghandi	injerencia
Topic 7	proyecto	chile	automotriz	macri	gobierno	generar	reforma	europa	sector	chileno
Topic 8	gobierno	méxico	crecimiento	político	rúa	reforma	inversión	fiscal	presidente	juez
Topic 9	gobierno	sindicato	mundo	economía	político	deber	chino	puerto	nacional	cambio
Topic 10	sistema	crisis	deuda	situación	deber	empresa	laboral	tema	crecimiento	plan

imagen 12.12 lista de frecuencias, dada por la nube de palabras del diario de La Nación dado en el periodo 2012-2016. Estas listas usaremos más adelante para multiplicar con las word embeddings.

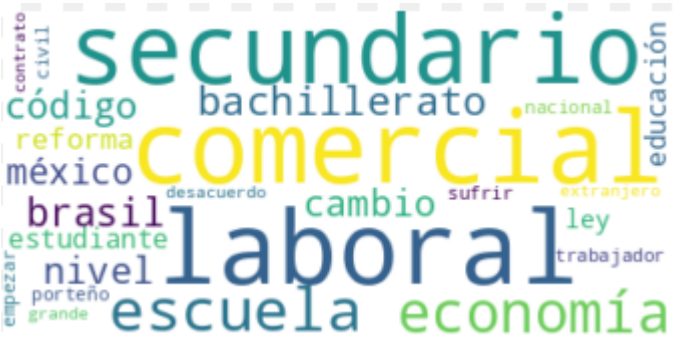


Imagen 12.13 se observa el tópico 1. De acuerdo a la imagen 12.12 la palabra laboral subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

Imagen 12.14 se observa el topic 5. De acuerdo a la imagen 12.12 la palabra faxconn subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	ministro	aumento	lavagna	salarial	gobierno	radical	sindical	precio	negociación	sueldo
Topic 2	gobierno	diputado	ley	proyecto	legislador	tratar	senado	político	pagar	norma
Topic 3	proyecto	ley	legislador	kirchner	creer	político	reforma_laboral	nacional	hablar	deber
Topic 4	gobierno	pinochet	modelo	mercado	fmi	chile	deber	dólar	economía	organismo
Topic 5	gobierno	reforma_laboral	piquetero	presidente	asegurar	político	ley	plan	rúa	derogación
Topic 6	ley	proyecto	gobierno	empresario	ministro	diputado	tema	laboral	congreso	presidente
Topic 7	trabajador	empresa	gobierno	proyecto	ley	fondo	empresario	laboral	social	deber
Topic 8	gobierno	fiscal	empresa	empresario	piquetero	recibir	corsiglia	deber	reforma_laboral	luis
Topic 9	kirchner	gobierno	político	sector	empresario	diputado	hablar	trabajador	presidente	dirigente
Topic 10	senador	juez	político	gobierno	presidente	pontaquarto	rúa	deber	genoud	piquetero

Imagen 12.15 lista de frecuencias, dada por la nube de palabras del diario de La Nación dado en el periodo 2003-2008. Estas listas usaremos más adelante para multiplicar con las word embeddings.

topic 1

imagen 12.16 se observa el topic 1. De acuerdo a la imagen 12.15 la palabra laboral subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

imagen 12.17 se observa el topic 1. De acuerdo a la imagen 12.15 la palabra laboral subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	jubilado	previsional	sistema	jubilación	político	aporte	trabajador	empresa	aumento	régimen
Topic 2	trabajador	empresa	proyecto	sector	gobierno	laboral	ley	condición	afip	registrado
Topic 3	sector	empresa	trabajador	economía	importación	industrial	español	capital	ley	actividad
Topic 4	trabajador	g20	proyecto	empresa	participar	gobierno	empresario	reunión	presidenta	cumbre
Topic 5	ingreso	trabajador	social	programa	desigualdad	hogar	sector	político	deber	laboral
Topic 6	empresario	gobierno	trabajador	sector	negociación	cgt	uia	empresa	salarial	salario
Topic 7	trabajador	conflicto	empresa	sindicato	ministerio	gremio	paro	petrolero	cartera_laboral	fuerza
Topic 8	político	social	senador	debate	deber	economío	rúa	pontaquarto	permitir	economista
Topic 9	fmi	europeo	fondo	economía	crisis	grecia	europa	deuda	euro	gobierno
Topic 10	crisis	nivel	desempleo	crecimiento	mercado	político	economía	trimestre	registrado	sector

magen 12.188 lista de frecuencias, dada por la nube de palabras del diario de La Nación dado en el periodo 2008-2012. Estas listas usaremos más adelante para multiplicar con las word embeddings.

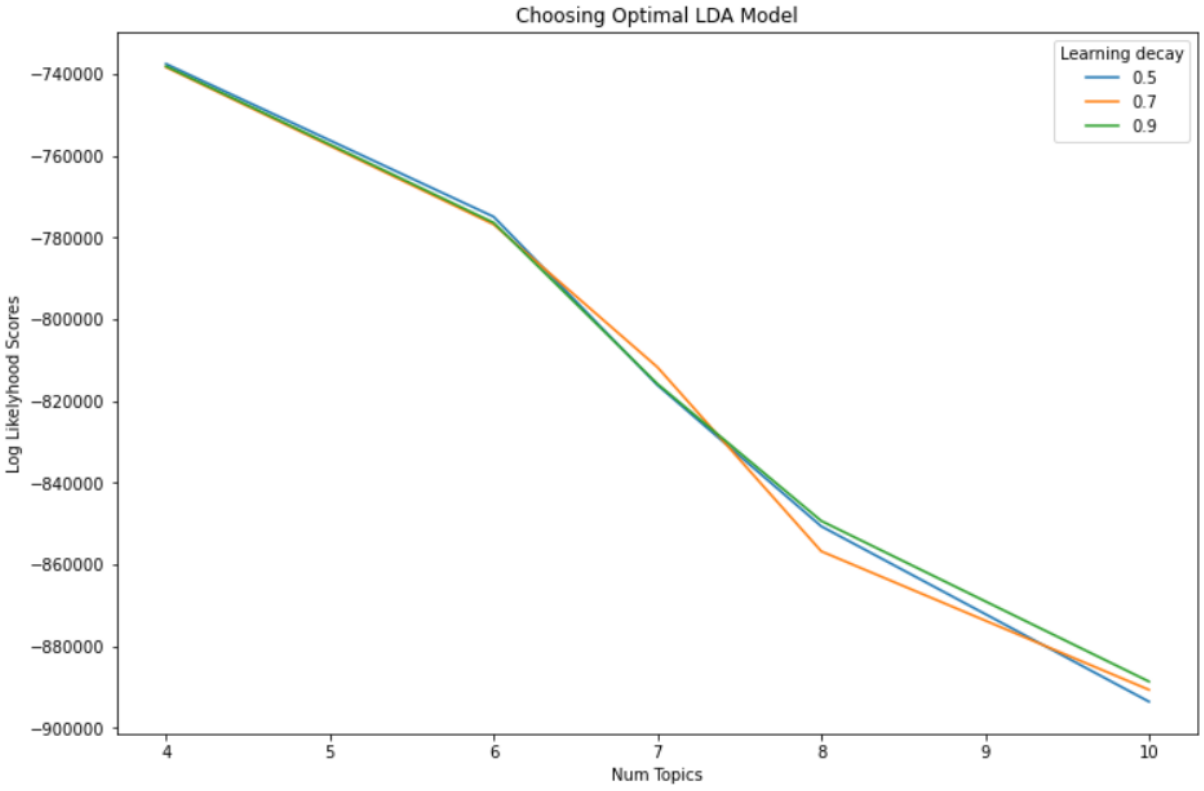
imagen 12.19 se observa el topic 1. De acuerdo a la imagen 12.18 la palabra laboral subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

imagen 12.20 se observa el topic 1. De acuerdo a la imagen 12.18 la palabra laboral subraya mayor cantidad de ocurrencias en el entrenamiento de LDA para este tópico particular.

Si bien el método estaba bien. Los resultados obtenidos eran insatisfactorios, estos se colapsaron, de tal manera que varios temas representaban uno sólo.

12.4 Gridsearch con sklearn.

Observando que nuestro experto de dominio no estuvo de acuerdo con los resultados obtenidos con gensim, ya que el mismo no proveía ciertas conjugaciones deseadas. Se decidió modelar en Sklearn.
Con sklearn se obtuvo más acceso a parámetros como alfa, beta, learning decay. Para esto usamos un grid search (Véase capítulo 7).
La siguiente imagen muestra experimentos del grid search para 10 topicos con alfa y beta de 0,6 y 0,4 respectivamente. Se observa que el número optimo es 10.



12.5 visualización de sklearn

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	gobierno	reforma_laboral	rua	deber	proyecto	sen	cgt	an	presidente	querer
Topic 2	gobierno	rua	cgt	reforma_laboral	an	politico	trabajador	deber	laboral	congreso
Topic 3	gobierno	reforma_laboral	rua	reforma	empresa	cgt	ley	legislador	alianza	gobernador
Topic 4	gobierno	cgt	reforma_laboral	empresa	an	sindical	diputado	rua	proyecto	trabajador
Topic 5	gobierno	an	politico	deber	reforma_laboral	reforma	rua	presidente	senador	trabajador
Topic 6	an	gobierno	trabajador	cgt	reforma_laboral	proyecto	legislador	negociacion	flamarique	moyano
Topic 7	clinton	pan	pri	prd	mexico	walsh	fox	mexicano	ypf	embajador
Topic 8	gobierno	reforma_laboral	cgt	proyecto	rua	laboral	trabajador	an	presidente	alianza
Topic 9	reforma_laboral	gobierno	cgt	trabajador	laboral	deber	politico	empresa	sindical	rua
Topic 10	gobierno	cgt	an	reforma_laboral	laboral	trabajador	proyecto	provincia	diputado	congreso

Topics periodo 2000-2003 la Nacion.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
Topic 1	generacion	trabajador	modernista	mejorar	gobierno	mar	reforma_laboral	chico	modernistas	alternativo
Topic 2	trabajador	gobierno	social	moyano	negociacion_colectivo	querer	an	senador	sindicato	reforma
Topic 3	an	trabajador	empresa	pyme	antiguedad	uia	derecho	pagar	pequen	vacacion
Topic 4	gobierno	biagi	reforma_laboral	politico	reforma	ministro	asesinato	asesinar	berlusconi	ataque
Topic 5	modernistas	gobierno	plan	chico	modernista	barcelona	apelar	dirigente	ministro	scajola
Topic 6	gobierno	reforma_laboral	ley	senado	proyecto	norma	an	bajar	trabajador	parecer
Topic 7	generacion	modernistas	chico	modernista	mar	barcelona	crear	votar	norma	mayorio
Topic 8	mar	barcelona	modernista	modernistas	generacion	salario	sindicato	reforma_laboral	simplemente	proyecto
Topic 9	bloque	uia	camara	plan	norma	legislador	empresario	ley	sector	laboral

Topics periodo 2000-2003 página 12.

si bien era prometedor el análisis con sklearn, ocurrían varias veces en distintos tópicos ciertas anormalidades como por ejemplo la palabra “an”, que no tenían significancia.
Otra peculiaridad encontrada fue en el comportamiento de LDA. Si bien LDA funciona bien con un corpus extremadamente mayor al nuestro, en nuestro caso nos encontramos con

pocos documentos y poca variabilidad en términos, esto sucede ya que nuestro tema era muy restringido.

12.6 Fasttext y similitud coseno

Una vez obtenido los modelos en LDA de cada tópico pasando por los distintos periodos de cada diario, se trabajó con Fasttext para obtener embeddings de cada diario y poder representar las dimensiones de cada tópico por periodo. El objetivo es lograr capturar las similitudes de cada diario con la función “distancia euclidiana”. (véase apartado 10.4)
La distancia representada en este caso, se observa en el siguiente gráfico.

Notamos que en el periodo 2003-2008 los artículos de ambos diarios tienen menor similitud que en 2009-2012. Con esto nos referimos a que hay mayor diferencia en los periodos 2003-2008 y menor diferencia en 2009-2012.

12.7 Top2vec

Nuestro último modelo que investigamos fue top2vec, (véase capítulo 10).
Este modelo tuvo más esperanzas y es el cual se estudió a profundidad, como ya se vió en el capítulo 10, top2vec modela mejor y con coherencia nuestros datos.

	Topic_0	Topic_1
Word_0	parecer	docente
Word_1	tema	maestro
Word_2	gobierno	decibir
Word_3	reforma_laboral	educativo
Word_4	querer	decibe
Word_5	perder	carpa
Word_6	manifestar	susana
Word_7	realidad	escuela
Word_8	momento	educacion
Word_9	armar	estatuto
Word_10	terminar	regimen_laboral
Word_11	discusion	sueldo
Word_12	volver	destinar
Word_13	capacidad	salarial
Word_14	esperar	chico
Word_15	plantear	ministerio
Word_16	cambio	porteno
Word_17	empresario	administrativo
Word_18	analizar	incremento
Word_19	posibilidad	provincia
Word_20	ex_ministro	secretaria
Word_21	etapa	provincial
Word_22	tipo	aplicacion
Word_23	cosa	provincio
Word_24	hablar	recurso
Word_25	presidente	automotor
Word_26	laboral	aumento
Word_27	criterio	gremio
Word_28	empresariado	incentivo
Word_29	dispuesto	presupuesto
Word_30	comenzar	aplicar
Word_31	deber	financiar
Word_32	politico	destinado
Word_33	amplio	salario
Word_34	escuchar	clase
Word_35	pensar	extender

Resultados de diario LaNacion en el lapso 2016-2019

	Topic_0	Topic_1	Topic_2
Word_0	storani	defender	sugerir
Word_1	listo	sistema	nivel
Word_2	presidente	problema	reducir
Word_3	decreto	dolar	flexibilidad
Word_4	alianza	empezar	economia
Word_5	banca	preocupacion	reduccion
Word_6	diputado	propuesta	disparar
Word_7	menem	tomar	posicion
Word_8	tratamiento	exterior	oficial
Word_9	alberto_pierri	mercado	cambio
Word_10	quorum	corto_plazo	mision_fondo
Word_11	abrir	caer	erman
Word_12	sesion	gobierno	economista
Word_13	sentar	credito	empresario
Word_14	decision	social	fuerte
Word_15	raul	crisis	compartir
Word_16	esto	venir	jose
Word_17	partidario	inversor	cartera_laboral
Word_18	camara	ingreso	empleo
Word_19	jefe	carlos_menem	roque_fernandez
Word_20	festejar	financiero	favor
Word_21	bloque	asiatico	ter_minassian
Word_22	texto	pasar	capital
Word_23	encontrar	externo	recibir
Word_24	menemismo	capital	gonzalez
Word_25	funcionario	inversion	encuentro
Word_26	legislador	financiar	empresa
Word_27	congreso	fiscal	desempleo
Word_28	lograr	materia	diputados
Word_29	lugar	empresario	privado
Word_30	opositor	fondo_monetario	fmi
Word_31	bloque_diputado	futuro	presion
Word_32	parlamento	tendencia	aspecto
Word_33	pese	crisis_asiatico	crecimiento
Word_34	oficialismo	advertencia	monetario
Word_35	discutir	construccion	actividad

Resultados de diario pagina12 en el lapso 2016-2019

	Topic_0	Topic_1	Topic_2	Topic_3	Topic_4	Topic_5	Topic_6	Topic_7	Topic_8	Topic_9
Word_0	cgt	tipo_cambio	regulacion	pt	paris	empleador	dni	riesgo_trabajo	asalariado	alumno
Word_1	dirigente	mediano_plazo	tecnologico	michel_temer	protestar	remuneracion	material	art	empleo	escuela
Word_2	central_obrero	competitivo	modalidad	temer	manifestacion	blanqueo	caracter	medico	monotributista	secundario
Word_3	gremio	competitividad	laboral	san_pablo	manifestante	regularizar	nombre	incapacidad	interanual	educativo
Word_4	cegetista	divisa	proteccion	rousseff	francia	multa	exceder	accidente	registrado	estudiante
Word_5	hugo_moyano	inversion	empleo	silva	afp	indemnizacion	destinado	juicio	trimestre	docente
Word_6	pablo_moyano	exportacion	trabajo	rio_janeiro	retirar	equivalente	respeto	administrativo	informal	educacion
Word_7	convocar	crecimiento	informalidad	lula	huelga	mensual	estilo	instancia	sector_privado	colegio
Word_8	hugo	agregado	autor	democratico	macron	monto	maria	litigiosidad	expansion	practicar
Word_9	miguel_pichetto	economista	contratar	brasileno	protesta	antiguedad	texto	comision	mercado_laboral	chico
Word_10	hector	mercado	contratacion	haddad	policia	registrado	ciudadano	judicial	indicador	formacion
Word_11	schmid	costo	flexible	jair_bolsonaro	movilizacion	pago	delito	cuerpo	privado	porteno
Word_12	casa_rosado	inflacion	profesional	congelamiento	combustible	salario_minimo	preso	enfermedad	comparacion	aprender
Word_13	gremial	desequilibrio	normativo	bolsonaro	enfrentamiento	beneficio	red	prestacion	contratacion	clase
Word_14	oficialismo	sostenido	humano	soborno	violencia	contribucion	espacio	obligatorio	cuarto	ministra
Word_15	camionero	financiar	empleador	corrupcion	socialista	aporte	grave	decreto	actividad	practica
Word_16	jorge_triaca	dolar	calidad	palacio	calle	sueldo	carta	determinar	numero	comunidad
Word_17	moyano	produccion	productividad	polemico	anunciado	aportar	calle	resolucion	desocupacion	evaluacion
Word_18	cta	precio	plataforma	partido	fuerza	trabajador	direccion	procedimiento	estimar	profesor
Word_19	carlos_acuna	deficit	formal	militar	ministra	compensacion	patria	fuero	formal	area
Word_20	triunvirato	deficit_fiscal	regular	tribunal	jornada	cesar	razon	indemnizacion	mensual	joven
Word_21	convocatoria	producto	estructura	apoyo	ciudad	empleado	lopez	adherir	creacion_empleo	ciudad
Word_22	senador	impuesto	colectivo	escandalo	masivo	contratar	exigir	ciudad_aires	caido	padre
Word_23	jorge_triaco	tasa_interes	creacion_empleo	comicio	detener	establecer	pueblo	monto	registrar	centro
Word_24	omar	monetario	cobertura	brasil	marchar	informalidad	prision	tribunal	crecer	implementacion
Word_25	juan_carlos	mejorar	mercado_laboral	mandato	provocar	vigente	iglesia	abogado	sector_publico	contenido
Word_26	sindicalista	mercado_interno	individual	condenar	pension	negro	vida	tramite	indec	calidad
Word_27	triaco	exportar	desempleo	liberal	convocado	tope	penal	previo	crecimiento	normal
Word_28	peronista	banco_central	incorporar	democracia	moreno	laboral	hermano	disponer	expectativa	vida
Word_29	gremialista	externo	trabajador	fernando	informar	empresa	nuevamente	grado	generar_empleo	universidad
Word_30	postura	infraestructura	formacion	recesion	prensa	prestacion	muerte	fallo	observar	quinto
Word_31	sindical	pbi	incentivar	asumir	plaza	calculo	entregar	sistema	cifra	estudiar
Word_32	senado	economia	adaptar	aliado	transporte	extra	libertad	aplicacion	semestre	tecnico
Word_33	peronismo	energia	tecnologia	presidencia	izquierda	salario	disponer	juez	construccion	completo
Word_34	kirchnerista	endeudamiento	oit	voto	derecho	formal	policia	regir	informalidad	capacitacion
Word_35	macri	cambiario	necesario	analista	orden	cobertura	justo	cobertura	expectativo	cultural

Resultados de diario LaNacion en el lapso 2016-2019

	Topic_0	Topic_1	Topic_0	Topic_1	Topic_2
Word_0	agregar	externo	cargo	trimestre	externo
Word_1	marco	economista	aplicar	centro	proceso
Word_2	recibir	crisis	reunir	documento	demanda
Word_3	pretender	tipo_cambio	sostener	caido	inversion
Word_4	declaracion	resultado	aceptar	registrado	divisa
Word_5	reunir	interno	participar	consumo	tipo_cambio
Word_6	aceptar	clave	compromiso	deterioro	asumir
Word_7	confirmar	historia	plata	contraccion	responder
Word_8	cargo	internacional	eliminar	retroceso	recesion
Word_9	reconocer	recesion	discutir	caida	economista
Word_10	asegurar	divisa	recibir	marcar	cambiario
Word_11	fuente	central	fuente	julio	similar
Word_12	tipo	interes	reclamar	incremento	economia
Word_13	advertir	demanda	declaracion	registrar	macroeconomico
Word_14	empezar	monetario	agregar	tension	tasa_interes
Word_15	denunciar	inversion	indicar	ubicar	resultado
Word_16	definir	cambiario	idea	interanual	internacional
Word_17	ofrecer	exportacion	federal	semestre	desarrollo
Word_18	incluir	tasa_interes	tipo	perdido	central
Word_19	ocurrir	macroeconomico	considerar	cifra	interno
Word_20	analizar	politica	modificar	asalariado	apuntar
Word_21	suceder	capacidad	tema	indicador	cuestion
Word_22	pasar	capital	contar	cantidad	extranjero
Word_23	situacion	escenario	respuesta	observar	clave
Word_24	intentar	apuntar	terminar	termino	dolar
Word_25	anunciar	gestion	camara	desocupacion	estrategia
Word_26	compromiso	devaluacion	limitar	actividad	capital
Word_27	similar	abrir	incluir	febrero	devaluacion
Word_28	discusion	decado	pasar	agosto	capacidad
Word_29	unidad	fondo_buitre	advertir	noviembre	comercial
Word_30	sostener	extranjero	carlos	crecer	crisis
Word_31	autoridad	dolares	discusion	junio	monetario
Word_32	responder	economia	traves	sector_privado	perder
Word_33	frente	banco_central	reclamo	periodo	definir

Resultados de diario pagina12 en el lapso 2016-2019

Este modelo lleva incorporado en su algoritmo la cantidad de Tópicos automáticamente, mientras que tomamos 50 palabras para representar cada tópico.

13 Conclusiones y trabajo Futuro.

En este trabajo se presentaron una serie de pasos para poder encontrar tópicos en dos diarios argentinos con el fin de poder, dado un texto (artículo), predecir su editorial de origen. Sin embargo, se presentaron problemas en dos de los modelos estudiados con LDA, concluimos que la cantidad de artículos no presentan

variabilidad y esto hacía que el modelo colapsara de tal manera que era imposible determinar tópicos.
En este punto descubrimos un nuevo modelo, y nos adentramos a investigarlo. Este modelo llamado top2vec, que ya lo presentamos en el capítulo 10 fue efectivo, nos dió resultados más coherentes a nuestra investigación por lo que se decidió continuar examinando los resultados exhaustivamente. Este inmenso labor de asistir a cada tópico, encontrar similitudes como así ubicarlos en el tiempo fue elaborado por el Doctor Andrés Matta, profesor de economía.
Este análisis que se nombrará a continuación, detalla con cierta certidumbre acontecimientos históricos de la economía del país.

13.1 Análisis del dataset

Periodo	1	2	3	4	5	6	7	Total artículos	Promedio artículos
	1995-1999	2000-2002	2003-2007	2008-2011	2012-2015	2016-2019	2020		
P12	13	24	130	505	582	194	6	1454	207,7
LN	748	399	302	130	155	1161	326	3221	460,1

Tabla 13.1. Hace referencia a la imagen 13.2

El relevamiento de artículos en cada período ha permitido detectar un primer aspecto que no ha sido identificado en la literatura especializada: los hallazgos muestran que el volumen de notas aumenta cuando las políticas del gobierno de turno coinciden con las líneas editoriales de cada periódico. En la Tabla 13.1 se puede observar que para Pagina12, el promedio entre los 7 períodos analizados es de 207,7 artículos y que este valor más que se duplica entre 2008 y 2015. Algo semejante pero en períodos opuestos sucede en el caso de La Nación, donde el promedio es de 460,1 notas y este valor es muy superior durante 1995-1999 y entre 2016-2019. Esta tendencia podría indicar que los medios tienen un rol procíclico, acompañando los ciclos de reformas alineadas con su propia perspectiva.

13.2 Análisis de los tópicos.

Este análisis ha permitido además identificar durante todo este lapso 45 tópicos, 20 para Pagina12 y 25 para el caso de La Nación. Un análisis de los mismos ha hecho posible clasificarlos en 7 grandes temáticas con cierta continuidad a lo largo del tiempo. En la siguiente Tabla se detallan los mismos y los términos más significativos de cada una. Antes de realizar un análisis más detallado de la distribución de estos tópicos, vale la pena señalar que, se observan temáticas fuertemente asociadas a períodos particulares (destacan en ello dos, referidas al ciclo de reformas de la década de 1990 y a su posterior judicialización) y otras que se extienden de manera más o menos continua durante todo el período analizado. Además, aunque no se han aplicado técnicas para detectar sentimientos, pueden

identificarse en algunas temáticas, ciertos términos y algunos tópicos que son característicos de cada periódico y de su posicionamiento frente al tema.

Temática de segundo nivel	**Núm. Tópico**	**Términos más significativos**	**Periódico y Período**
Sanción y derogación de leyes de 1990	1	Quorum, decreto, menemismo	P12.1
	2	Justicialista, sanción, senador, Flamarique, derogación, Moyano	P12.2
	3	Flamarique, De la Rua, desocupado, justicialista, Moyano, emergencia	P12.3
	4	Derogación, FMI, CGT, plan, empresario, mercado, ley, derecho	P12.4
Proyectos de reforma/flexibilización específicos	5	Indemnización, despido, preaviso, limitar, estatuto, accidente, enfermedad, Pyme, competitividad	P12.2
	6	Beneficio, limitar, pequeño-mediano, Pyme, despido, accidente, estatuto, indemnización, período prueba, indemnización, competitividad, enfermedad	P12.2
	7	Empleo, fiscal, ganancia, impuesto, indemnización, inflación, salario	P12.7
	8	Despedir, derecho, ganancia, impuesto, indemnización, inflación	P12.7
	9	Abogado, accidente, ART, comisión, enfermedad, fuero, indemnización, incapacidad, litigiosidad, médico,	LN.6
	10	Antigüedad, aporte, blanqueo, despido, indemnización, informalidad, regularizar, tope	LN.6
	11	Regulacion, tecnológico, Empleo, flexible, contratación, creación empleo, empresa, formación, informalidad, productividad, tecnología, , trabajador	LN.6
	12	Construcción, contratación, creación empleo, crecimiento, formal, informalidad, monotributista, registrado	LN.6
	13	Contrato, plazo, salario, reducción, seguro, beneficio, dialogo, empresario, impuesto	LN.3

Oposición a proyectos de reforma/flexibilización	14	Rechazar, plan, UIA, banquero,	P12.2
	15	CGT rebelde, convertibilidad, salario, sector, CGT, costo, producir, capacidad	P12.2
	16	Camionero, CGT, CTA, gremio, Moyano, metalúrgico, negociación, portuario, UIA, UOM	P12.4
	17	Denunciar, Macri, Secretario, unidad	P12.5
	18	Moyano, peronista, oponer, justicialismo, opositor, peronismo, proyecto ley, cuestionar, rechazar	LN.2
	19	Aprobar, derogar, indemnización, Moyano, proyecto, Recalde, sindicalista	LN.3
	20	Camionero, CGT, CTA, Moyano, peronismo, sindical	LN.6

Marco económico y fundamentos de las reformas	21	Flexibilidad, capital, empresa, desempleo, cambio	P12.1
	22	Fondo monetario, financiero, capital, inversión, plan	P12.1
	23	FMI, flexibilización, compeititividad, salario, CGT rebelde, desocupación, sindical, gremial	P12.2
	24	Ajuste, capital, economía, endeudamiento, fondo buitre, inversión, mercado, recesión	P12.6
	25	Capacidad, complejo, economía, mercado, nacional, política, reducción, local, internacional, debilidad	P12.5
	26	Incentivo, informal, patronal, plan, rentabilidad, sector, segmento, sindicato, gremio	P12.6
	27	Costo, inversión, mercado, limitar, banco, financiero, estabilidad, enonomía	LN.2
	28	Empresariado, cambio, profundo, Menem	LN.1
	29	América latina, difícil, crecer, deuda, gasto, organismo, pagar, riesgo	LN.3
	30	Costo, empleador, servicio, personal, social, plan	LN.4
	31	Salario, servicio, costo, sistema, demanda, plazo, pagar, salario	LN.5

	32	Tipo cambio, Bajar impuesto, bajar inflación, competitividad, costo, crecimiento, déficit, eficiencia, exportar, gasto, inversión, producción	LN.6
	33	Emergencia, incertidumbre,	LN.7

Judicialización de las reformas	34	Coima, maniobra, delito, escandalo, juicio, soborno, Flamarique	P12.3
	35	Coima, soborno, De la Rua, senado, juicio	P12.5
	36	Pontaquarto, judicial, SIDE, soborno, Senado	LN.3

Cambio tecnológico y educativo	38	Educación, régimen laboral, incentivo, salario	LN.1
	39	Tecnológico, capacitación, docente, educación, formación, pasantía, calidad	LN.6

Reformas internacionales	40	Chirac, socialista, francés	LN.3
	41	Temer, Rousseff, soborno, recesión, gasto público, congelamiento	LN.6
	42	París, protestar, violencia, sindicato, huelga	LN.6
	43	PP, PSOE, Sanchez, dialogo,	LN.7
	44	Zapatero, Alemania, desempleo, FMI, Grecia, rescate, reducción	LN.4
	45	Francia , crecer, despido, financiero, inflación, Madrid, plan, programa, reducir	LN.5

Tal como se adelantó previamente, el análisis permite identificar algunas particularidades en el tratamiento de las temáticas en cada periódico, las que pueden ser asociadas a agendas y marcos interpretativos diferentes. Generalizando, La Nación orientada a reformas que se asocian a la agenda empresaria y Página 12 a intereses de los sectores sindicales. En este sentido, el análisis permite ratificar algunos hallazgos realizados con técnicas cualitativas tradicionales (supervisadas) (Coscia, 2020; Coscia y Perbellini, 2020) pero también revela otros aspectos no mencionados en la literatura:

a) En La Nación se identifica una temática exclusiva, que asocia la necesidad de reformas en el trabajo al cambio tecnológico, que a su vez, debe vincularse a mejoras en el sistema educativo y la capacitación. Estos temas son tratados en Pagina12 pero sólo aparecen con la relevancia de un tópico en LN justamente durante los dos períodos de gobierno afines a la línea editorial. En el primer caso, se asocian a las discusiones de la década de 1990 sobre los salarios docentes mientras que entre 2016 y 2019 claramente vinculadas a los proyectos para generar reformas flexibilizadoras y regímenes especiales como las pasantías.

b) Otra temática que aparece con volumen y relevancia solamente en La Nación es la agenda de reformas internacionales. De manera prácticamente sostenida desde 2003 en adelante, se han tratado las reformas realizadas en Brasil, España, Francia, y Grecia, muchas de ellas con fuerte resistencia sindical. Un análisis de los artículos permite observar que en general esta agenda tiende a mostrar a estos casos como ejemplos a seguir (Coscia y Perbellini, 2020). Esto no es algo que se pueda constatar solamente con los términos identificados pero este estudio capta la importancia de estos tópicos y da pie a futuros análisis de contenido.

c) Por su parte, la única temática que podemos decir adquiere entidad propia de Página 12 se relaciona con el tratamiento legislativo y los intentos de derogación de las leyes laborales más importantes de la década de 1990. Estos tópicos aparecen en los tres primeros períodos e incluyen la dificultosa aprobación de la ley 25.013 en 1998 y la 25.250 en 2000 y sus intentos de derogación por parte de distintos sectores sindicales hasta lograrlo en 2004 mediante la ley 25.877.

d) No obstante lo dicho en el apartado anterior, en LN existe una temática recurrente que se comparte con P12 que puede asociarse a la citada pero puede enmarcarse de modo más genérico como de oposición a los distintos proyectos de reformas “flexibilizadoras”. En el caso de P12, se pone el acento en los actores opositores (CGT, CTA, Camioneros, Moyano) y algunos impulsores de las reformas (UIA) y en términos como “rechazar”, “denunciar”. En el caso de LN aparecen similares actores pero se tiende a identificarlos más claramente su sector partidario “peronista”, “justicialismo”, “peronismo”.

e) Otra temática tratada por ambos periódicos se vincula con los proyectos de reforma y las reformas sobre temáticas específicas (en general orientadas a la “flexibilización”) de los derechos laborales. En el caso de P12 aparece claramente en el período 2000-2002 asociando el término “limitar” a las “indemnizaciones”, “accidentes”, “enfermedad”, “período de prueba”, etc y luego en 2020, vinculado nuevamente a las indemnizaciones y también al salario (aunque en este caso vinculado al término “derecho”). En el caso de LN, los tòpicos aparecen con mayor evidencia en el período 2016-2019 asociando términos semejantes como “indemnización”, “enfermedad”, “ART” pero a otras connotaciones que podrían

leerse como positivas (“regularizar”, “flexible”, “productividad”, “creación de empleo”) y en cambio negativas hacia la situación actual y particularmente su “litigiosidad”. Estos aspectos surgen en el marco de intentos de reformas sectoriales (“tecnología”, “construcción”, etc) y de proyectos de ley como el “blanqueo laboral”.

f) Particularmente interesante resulta otra temática que atraviesa todo el período bajo análisis y que incluye el marco económico y los fundamentos de las reformas. En LN los términos más comunes se asocian a la agenda de los agentes económicos (“empresariado”, “mercado”) en la que se observa al trabajo desde el punto de vista de la “inversión” y el “costo” y en el marco de otras medidas asociadas a la “competitividad”, el “crecimiento”, la “eficiencia”, el “gasto”, los “impuestos”, etc.

Este mismo eje aparece en P12 pero en este caso, los actores que aparecen motorizando las reformas y sus motivaciones son otros. En primer lugar el “Fondo Monetario”, pero también los “fondos buitre”, el “capital” y se asocia a términos como “ajuste”, “flexibilidad”, “desempleo”, “recesión” y “endeudamiento”. Una cuestión particular es que estos tópicos aparecen con estas connotaciones en los períodos de gobiernos no afines a la línea editorial de este medio. En cambio, entre 2008 y 2015 aparecen dos tópicos que destacan la necesidad de realizar reformas laborales para solucionar otros problemas tales como la “informalidad” o el “ingreso” y en este caso se apunta resolver cuestiones estructurales como la puja distributiva y la debilidad del aparato productivo: “debilidad”, “capacidad”, “complejo”, “sector”, “rentabilidad”.

Lo mencionado en los párrafos precedentes puede complementarse al observar cuál es el grado de aparición de algunos términos entre los más relevantes de cada tópico y en cada medio. Allí se observa claramente el rol antagónico de ciertos actores del mundo sindical y antagonistas como el FMI en P12 y en cambio en LN el rol en el discurso para impulsar la “regularización” del “mercado laboral” de la “tecnología”, el “crecimiento”, el sector “privado”, la “educación”, o la disminución de la “litigiosidad” asociando al “peronismo” en general con el principal opositor a estas medidas.

13.3 Futuro trabajo

Como trabajo futuro, proponemos probar en otros modelos, obtener representaciones de words embeddings distintos y poder analizar comparando las distancias euclidianas propuestas en este trabajo. Si bien trabajamos con dos modelos (LDA y top2vec), estos eran similares. por lo que ambos, en el fondo,

trabajaban con LDA.

Otro aspecto importante, es justamente, lo explicado en el capítulo anterior, de obtener un análisis por sentimiento, obteniendo así un sentido a cada tópico o analizar en el tiempo las diferentes opiniones de las dos editoriales.

Consideramos necesario adaptar en otros modelos embeddings para procesar los artículos, y con dichos resultados lograr una comparación con fasttext.

Bibliografía:

[1] Isasi Vinuela, Pedro y Galvan Leon, Inés M. Redes de neuronas artificiales. Un enfoque práctico. Madrid: Pearson educación, S.A., 2004.
[2] Mark Steyvers University of California, Irvine "Probabilistic Topic Models"
[3] Lucas Ou-Yang, "Article scraping & curation", https://github.com/codelucas/newspaper/tree/python-2-head
[4] K.F. Shaalan, M.R. Girgis and Chia-Hui Chang. A Survey of Web Information Extraction Systems. 2006
[5] Jason Huggins at ThoughtWorks in Chicago. "The selenium documentation", 2004 https://www.selenium.dev/documentation/
[6] Lucas Ou-Yang 2019 , "Newspapers 3k " https://newspaper.readthedocs.io/en/latest/
[7] GARCÍA-AMARO, Jesús Fidencio Martinez Rodriguez , Revisión de técnicas de pre-procesamiento de textos para la clasificación automática retweets español , 2017
[8] A Linguistic Failure Analysis of Classification of Medical Publications: A Study on Stemming vs Lemmatization, Giorgio Maria Di Nunzio Dept. of Information Engineering University of Padua, Italy. Federica Vezzani Dept. of Languages and Literary Studies University of Padua, Italy http://ceur-ws.org/Vol-2253/paper45.pdf
[9] Christopher D. Manning, Hinrich Schiitze Foundations of Statistical Natural Language Processing 1999 Massachusetts Institute of Technology.
[10] Christopher Manning & Hinrich Schutze, MIT Press (1999). "Foundations of Statistical Natural Language Processing", https://dl.acm.org/doi/abs/10.1145/601858.601867
[11] Kenneth Ward Church and Patrick Hanks (March 1990). "Word association norms, mutual information, and lexicography". https://aclanthology.org/J90-1003/
[12] Mary James, Book series, "Unsupervised and Semi-Supervised Learning" https://www.springer.com/series/15892
[13] John Wiley & Sons Ltd. "De Finetti. Theory of probability" , Chichester, 1990. Reprint of the 1975 translation.
[14] David M. Blei , Andrew Y., Michael I. Jordan. " Latent Dirichlet Allocation" . Journal of Machine Learning Research 3 (2003).
[15] Sang-Woon Kim & Joon-Min Gil. "Research paper classification systems based on TF-IDF and LDA schemes" (2019).
[16] Haaya Naushan "Topic Modeling with Latent Dirichlet Allocation" (2020), https://towardsdatascience.com/topic-modeling-with-latent-dirichlet-allocation-e7ff75290f8
[17] Ben Greer "Sklearn LDA vs. GenSim LDA". (2018) https://medium.com/@benzgreer/sklearn-lda-vs-gensim-lda-691a9f2e9ab7
[19] Carson Sievert, Kenneth E. Shirley "LDAvis: A method for visualizing and interpreting topics
[20] In 2011, I moved Gensim's source code to Github and created the Gensim website. In 2013 Gensim got its current logo, and in 2020 a website redesign. https://github.com/RaRe-Technologies/gensim
[21]This project was started in 2007 as a Google Summer of Code project by David Cournapeau. Later that year, Matthieu Brucher started work on this project as part of his thesis. https://scikit-learn.org/

[22] Z. S. Harris. Distributional structure. WORD, 1954.
[23] P.-N. Tan, M. Steinbach & V. Kumar, "Introduction to Data Mining", Addison-Wesley (2005),
[24] “Top2Vec: Distributed Representations of Topics.” Dimo Angelov (2020)
[25]“Jordan L Boyd-Graber and David M Blei.” Syntactic topic models. In Advances in neural information processing systems” (2009).
[26] S. Syed and M. Spruit. “Full-text or abstract? examining topic coherence scores using latent dirichlet allocation” (2017).
[27] Kai Yang, Yi Cai, Zhenhong Chen, Ho-fung Leung, and Raymond Lau. “Exploring topic discriminating power of words in latent dirichlet allocation” (2016).
[28] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, Martin Vetterli “Euclidean Distance Matrices: Essential Theory, Algorithms and Applications”, Aug 2015 (v2)