CONVENCIÓN
“ TITLEEEE”

XXXXXXXXXXXXXXXXX

Método para la detección de tópicos en la red social de mensajería instantánea de la UCI
(Topic detection in Jabber instant message network)

Resumen

El uso de aplicaciones y sistemas dentro de estos entornos genera un enorme cúmulo de información y datos que son desaprovechados a causa de que la misma no está estructurada, por lo que de nada sirve tener un gran volumen de datos donde no se puede extraer el valor de la información. La detección de tópicos consiste en descubrir las estructuras semánticas subyacentes en documentos y buscar similitudes entre textos. Esta tiene como propósito colocar de forma automática, documentos dentro de un número fijo de categorías (temas o clases) predefinidas, en función de su contenido. El presente trabajo propone desarrollar un método que permita identificar tópicos en los mensajes emitidos en la red de mensajería instantánea de la Universidad de las Ciencias Informáticas (UCI). Se describió el procedimiento que sigue el método presentado, se analiza el funcionamiento del modelo Latent Dirichelt Allocation (LDA) para la detección y se realizaron validaciones en aras de verificar la calidad de la solución. Como resultado final se obtienen visualizaciones de los tópicos, así como las palabras claves representativas de cada uno, lo que puede ser usado para la toma de desiciones.

Palabras clave: detección de tópicos, Latent Dirichlet Allocation, redes de mensajería instantánea, preprosesamiento, visualizacion

1. Introducción

Las redes sociales actúan como un punto de encuentro donde millones de usuarios pueden publicar y compartir opiniones, información o simplemente trivialidades sobre todo tipo de temas. Poder analizar y comprender toda esa información pública se está convirtiendo en uno de los principales objetivos por parte de empresas y organizaciones, que ven en estos lugares una fuente de información para analizar qué se dice sobre su área de influencia (Vilares Calvo, 2014).

Usualmente al escuchar el término red social se piensa en las grandes y conocidas redes sociales Twitter, LinkedIn, Facebook o Instagram, pero estas redes representan solo una pequeña minoría del número de

.

mecanismos de interacción social presentes en la web. Por lo que se hace necesario resaltar que debido a que se basan en la interacción entre diferentes individuos y por tanto, tienen una naturaleza social, los correos electrónicos, blogs, fórums o los sistemas de mensajería instantánea, entre otros, también son considerados redes sociales (Aggarwal, 2015).

La mensajería instantánea, es una forma de comunicación en tiempo real entre dos o más personas, basada en mensajes de texto, a través de dispositivos conectados a una red, lo cual ha facilitado la comunicación entre las personas. Sin embargo, la mensajería instantánea es un arma de doble filo y podría ser mal utilizada para el intercambio de información ilegítima, influir en el comportamiento y emociones de las personas, incentivar o cometer actos delictivos. Es por ello que identificar, procesar y presentar los temas de conversación dentro de las interacciones de los usuarios en un formato que facilite su comprensión y análisis, ha sido objeto de atención por parte de la comunidad de investigadores y administradores de este tipo de sistemas (Brun and Senso, 2004).

En los sistemas de mensajería instantánea de redes institucionales, el poder determinar y descubrir las temáticas y/o tendencias de las conversaciones entre los usuarios de dicho sistema, cobra mayor importancia aún, sobre todo en redes institucionales de centros educativos (Ng, 2016). En este caso se encuentra la Universidad de las Ciencias Informáticas (UCI), en la cual existe una red de mensajería instantánea (comúnmente conocida como Jabber), de la cual son miembro (o pueden interactuar en ella) todos los estudiantes y profesionales (trabajadores y especialistas) de la universidad.

Es muy interesante tener en cuenta también que los usuarios de las redes de mensajería instantánea suelen escribir a sus contactos cuando algo no anda bien, es decir, para quejarse al respecto de un servicio, producto, evento o persona como forma de desahogo, y por lo general, tiende a compartirlo con más de un contacto, y estos a su vez con sus contactos. Esto significa, que estas conversaciones son un gran termómetro que permiten medir el estado de opinión de los usuarios, así como identificar los temas que más preocupan a la comunidad e identificar posibles rumores o comportamientos no permitidos por los reglamentos internos de la universidad.

Diariamente la red de mensajería instantánea de la UCI genera un gran volumen de datos. Estos datos son analizados con herramientas que no distinguen entre la información relevante y no relevante, por lo que se invierte demasiado tiempo en el análisis de dicha información, lo que trae consigo la pérdida de la oportunidad de prevenir actividades ilegítimas en función de los reglamentos internos de la institución. Además, dichas herramientas limitan la obtención e identificación de los temas tratados en las conversaciones entre los usuarios. Por otra parte, el carácter subjetivo del análisis realizado, y la carencia del personal necesario para realizar dicha tarea en su totalidad, hace que los resultados conseguidos sean inexactos e incompletos, y se ve afectada la fidelidad y el alcance de los mismos. A partir del análisis anterior es posible afirmar que los resultados obtenidos son ineficaces a pesar de los altos costos en tiempo y recursos destinados.

A partir de la situación anteriormente descrita se define el siguiente **problema a resolver**: el modo en que se analizan los datos generados por el sistema de mensajería instantánea de la Universidad de las Ciencias Informáticas limita la identificación proactiva de ideas, opiniones o expresiones que se repiten con frecuencia en determinadas circunstancias, usualmente conocidas como tópicos.

El **objetivo general**: desarrollar un método que permita detectar tópicos en los mensajes emitidos en la red de mensajería instantánea de la UCI.

El presente documento cuenta con una estructura de tres capítulos, los cuales abordan los siguientes temas:

- Capítulo 1 Fundamentación teórica sobre el análisis de datos en redes de mensajería instantánea: Se definen los conceptos más relevantes relacionados con las redes sociales, minería de texto, y algoritmos de detección de tópicos, así como el Proceso de Descubrimiento del Conocimiento con las fases que lo componen.
- Capítulo 2 Método para la detección de tópicos en la red de mensajería instantánea de UCI: se definen las tecnologías y herramientas a utilizar en la construcción de la solución. Se presenta un método para determinar tópicos en la red de mensajería instantánea de la UCI a partir de los datos generados por el sistema.
- Capítulo 3 Validación del método implementado: se realiza la validación del método desarrollado mediante varios criterios para determinar que la solución propuesta responde satisfactoriamente al objetivo de la investigación.

2. Metodología

2.1. Presentación de la solución

Para el cumplimento del objetivo general del presente trabajo se deben detectar tópicos a partir de los mensajes de la red de mensajería instantánea de la UCI. El método propuesto sigue el procedimiento que se describe a continuación:

1. Carga y procesamiento de los datos presentes en el conjunto de datos de estudio.
2. Aplicación del modelo LDA a partir de los datos procesados.
3. Detección e interpretación de tópicos.

En la Fig 2.1 se muetra la estructura del metodo antes mencionado.

2.2. Carga y procesamiento de datos

2.2.1. Carga de datos

Para el desarrollo del presente trabajo se tomó como corpus un registro de mensajes de la red de mensajería instantánea de la Universidad de las Ciencias Informáticas almacenado en un archivo de extensión .csv, donde las columnas, que contienen los atributos del mensaje están separadas por comas, y las filas, que contienen un mensaje, por saltos de línea.

Cada mensaje tiene la siguiente estructura:

Conversación, Emisor, Cemisor, Receptor , Creceptor, Tiempo, Contenido

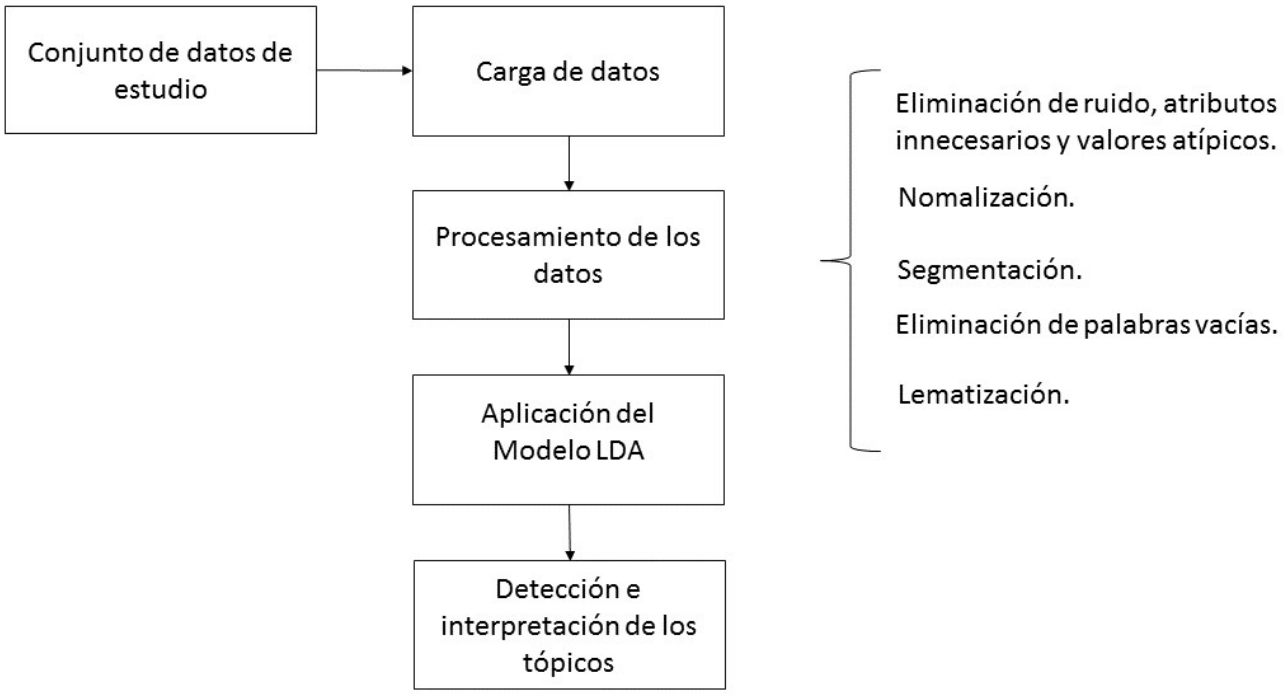

Figura 1. Estructura del metodo propuesto. Fuente: elaboración propia.

```
107923782,æ¬ð¥á¬,jabber.uci.cu/1a9efae7,xä€¡¬€ç®,jabber.uci.cu/f96e751e,1576026914488,"tengo metodologia  objetivos especificos"
```

Figura 2. Estructura de un mensaje.

- Conversación: es un número que identifica una conversación en el dominio.
- Emisor y Receptor: son cadenas de texto únicas que idetifican al usuario que emite y recibe el mensaje respectivamente.
- Cemisor y Creceptor: son cadenas de texto que identifican el cliente de mensajería instantánea del que emite y recibe el mensaje respectivamente.
- Tiempo: es el instante de tiempo en milisegundos en que se envía el mensaje.
- Contenido: es una cadena de texto con el contenido del mensaje.

Para desarrollar correctamente la solución que se propone se hace necesario cargar este registro de mensajes en una estructura que sea fácil y rápida de manipular ya que, una vez que los datos sean correctamente cargados va a ser necesario procesarlos. El manejo de este tipo de estructuras se lleva a cabo, usualmente, con lenguajes de programación de alto nivel. En la presente investigación se decidió hacer uso del lenguaje de programación Python en su versión 3.7.3 debido al gran número de librerías que posee para la manipulación eficiente de grandes volúmenes datos.

La librería de Python, Pandas, facilita el proceso de análisis de grandes volúmenes de datos de manera eficiente, por tal motivo, se decidió hacer uso de la misma en la presente investigación cargando el conjunto de datos de entrada en un objeto DataFrame de Pandas. Un objeto DataFrame de Pandas es una estructura de datos de alto rendimiento con la que es posible manejar fácilmente datos tabulares para el análisis de datos. En estos objetos cada una de las filas representa un registro. Mientras que en cada una de las columnas representan variables y, en este caso se define un nombre para cada una de estas columnas

puesto que no poseen en la fuente original.

```
import pandas as pd
def load_data(path):
    data = pd.read_csv(path, names=['id_conversation', 'id_sender', '
       id_sender_client', 'id_receiver', 'id_receiver_client', 'time',
        'message'], nrows=100000)
    return data
```

Figura 3. Función utilizada para cargar los datos de estudio.

Una vez cargado el registro de mensajes se inicia la fase de preprocesamiento de los datos, la etapa más importante en cualquier actividad de minería de textos ya que la calidad de la minería depende directamente de la eficiencia de esta fase Andrade Alvarado et al. (2015).

	id_conversation	id_sender	id_sender_client	id_receiver	id_receiver_client	time	message
20	107915949	€ä×çßçþ¬	jabber.uci.cu/4db432fe	»¥þç€¬€¥é»	jabber.uci.cu/1a543b23	1.574226e+12	tu si tienes la apk de instagram
21	107915947	€ä×çßçþ¬	jabber.uci.cu/4db432fe	¬é¬¶¬€ç€	jabber.uci.cu/df36ffa9	1.574226e+12	a ese me pincha
22	107915948	¬&x¬þçé	jabber.uci.cu/2cb78369	¬&x¬þçé	jabber.uci.cu	1.574226e+12	9204 1299 7382 2267
23	107915947	¬é¬¶¬€ç€	jabber.uci.cu/df36ffa9	€ä×çßçþ¬	jabber.uci.cu/4db432fe	1.574226e+12	t lo envio x dukto
24	107915948	¬&x¬þçé	jabber.uci.cu/2cb78369	¬&x¬þçé	jabber.uci.cu/2cb78369	1.574226e+12	bpa
25	107915947	€ä×çßçþ¬	jabber.uci.cu/4db432fe	¬é¬¶¬€ç€	jabber.uci.cu/df36ffa9	1.574226e+12	pera deja pedirle el ip al choco que yo stoy w...
26	107915947	¬é¬¶¬€ç€	jabber.uci.cu/df36ffa9	€ä×çßçþ¬	jabber.uci.cu/4db432fe	1.574226e+12	ok
27	107915948	¬&x¬þçé	jabber.uci.cu/2cb78369	¬&x¬þçé	jabber.uci.cu/2cb78369	1.574226e+12	9204 0699 9327 0321 bandec
28	107915946	ü<x¬þé¥ááæ	jabber.uci.cu/8c4e42e5	€ä×çßçþ¬	jabber.uci.cu/4db432fe	1.574227e+12	me debes un cerbero
29	107915946	€ä×çßçþ¬	jabber.uci.cu/4db432fe	ü<x¬þé¥ááæ	jabber.uci.cu/8c4e42e5	1.574227e+12	jaajaj ok te sirve uno negro?

Figura 4. Registro de mensajes cargados en objeto DataFrame de Pandas.

2.3. Preprocesamiento de datos

El preprocesamiento abarca todas aquellas técnicas de análisis de datos que permiten mejorar la calidad de un conjunto de datos de modo que las técnicas de extracción de conocimiento puedan obtener mayor y mejor información. Estos tratamientos conllevan a analizar las palabras del lenguaje y conseguir obtener la información relevante y filtrar la que no tiene significado semántico Andrade Alvarado et al. (2015). Dentro del preprocesado de texto, se procede a la eliminación de palabras vacías, lematización, normalización y segmentación del texto. Con estos pasos cada documento de la clase seleccionada se convierte a una representación compacta adecuada para el algoritmo de aprendizaje no supervisado, LDA.

2.3.1. Eliminación de ruido, atributos innecesarios y valores atípicos

Inicialmente se descartan las columnas (atributos) innecesarias o irrelevantes para la solución propuesta, se manejan los valores nulos y se elimina el ruido o valores atípicos. Para ello se crea un nuevo DataFrame de Pandas en donde se insertarán los datos luego de ser tratados debidamente.

Utilizando el juicio de expertos se llegó a la conclusión de que columnas como Cemisor y Creceptor eran irrelevantes para la propuesta de solución. También se determinó que la presencia de mensajes con un cambio de línea en su atributo Contenido generaba un nuevo registro con valores nulos y ruido en el conjunto de datos de estudio por lo que debieron de ser tratados.

Luego de descartar atributos innecesarios y tratar correctamente los valores atípicos y el ruido en el set de datos, se procede a la limpieza del texto que servirá de entrada al algoritmo (corpus).

2.3.2. Normalización

El proceso de normalización es muy importante en el preprocesamiento de un texto, ya que impide que el programa tome como palabras distintas las palabras que son sintácticamente iguales Andrade Alvarado et al. (2015). Para el caso de este trabajo se pasó todo el texto a minúsculas, se eliminaron todos los acentos y diéresis presentes en los mensajes (documentos). Además, se tuvo en cuenta que por lo general las conversaciones que se llevan a cabo en este sistema de mensajería son, en su mayoría, informales por lo que se decidió eliminar caracteres repetidos consecutivamente y patrones repetidos dentro de una misma cadena.

```
import unicodedata
import itertools
def accent_fold(s):
    return ''.join(c for c in unicodedata.normalize('NFD', s)
                   if unicodedata.category(c) != 'Mn')
def deleteConsecutiveChars(text):
    words=[]
    for word in text:
        word = ''.join(ch for ch, _ in itertools.groupby(word))
        word = re.sub(r'(.+?)\1+', r'\1', word)
        words.append(word)
    return words
```

Figura 5. Funciones utilizadas para normalizar el texto.

2.3.3. Segmentación

La segmentación es el proceso de separar el texto en unidades, denominadas *tokens*. En este caso se tokenizaron las palabras que componen cada uno de los contenidos de los mensajes del registro, haciendo uso de la librería de Python spaCy, una librería de código abierto para el procesamiento avanzado de lenguaje natural.

Utilizando también el juicio de expertos se determinó que los tokens que poseen una longitud menor a cuatro y los tokens cuya longitud excede a 20 caractéres no aportan información relevante al descubrimiento de tópicos por lo que fueron descartados.

2.3.4. Eliminación de palabras vacías

Las palabras vacías (*stopwords*) son palabras que son muy frecuentes y que por lo general no contienen información importante para la extracción de información, por ejemplo: pronombres, preposiciones, conjunciones, artículos, etc.

```
from spacy.lang.es import Spanish
parser = Spanish()
def tokenize(text):
    lda_tokens = []
    tokens = parser(text)
    for token in tokens:
        if token.orth_.isspace():
            continue
        else:
            lda_tokens.append(token.lower_)
    return lda_tokens
```

Figura 6. Función utilizada para segmentar el texto.

Estas palabras deben ser excluidas del texto porque se mejora la eficiencia y efectividad de los sistemas de minería de texto. Además de agregar ruido y aumentar el tamaño de la representación.

Empleando el Kit de Herramientas de Lenguaje Natural (NLTK por sus siglas en inglés *Natural Languaje Tool Kit*) de Python, se le dió tratamiento a estas palabras.

En adición al diccionario de palabras vacías que provee NLTK se elaboró un diccionario propio con ciertos términos que no son de interés para el negocio, por tanto, todo los términos que se encontraran en ese diccionario fueron descartados.

```
import nltk
stopwords = set(nltk.corpus.stopwords.words('spanish'))
def isStop(token):
    if token not in stopwords:
        return token
```

Figura 7. Función utilizada para eliminar las palabras vacías.

2.3.5. Lematización

Por lematización nos referimos al proceso de remover los sufijos para reducir una palabra a su lema o raíz.

Frecuentemente, una palabra no aparece exactamente en un documento, pero sí alguna variante gramatical de la misma como plural, gerundios, sufijos de tiempo verbal, etc. Este problema puede resolverse con la sustitución de las palabras por su raíz.

Haciendo uso de la librería Wordnet que provee NLTK fue posible lematizar todos los tokens.

Terminado el procesamiento de los datos, el texto está limpio y listo para ser empleado como entrada al algoritmo.

```
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer
def get_lemma(word):
    lemma = wn.morphy(word)
    if lemma is None:
        return word
    else:
        return lemma

def get_lemma2(word):
    return WordNetLemmatizer().lemmatize(word)
```

Figura 8. Función utilizada para lematizar los tokens.

2.4. Aplicación del modelo LDA

Se decidió aplicar la implementación del algoritmo LDA que posee la biblioteca Gensim de Python, diseñada para el modelado de temas no supervisado, el procesamiento de lenguaje natural y el manejo de grandes colecciones de texto.

Las dos entradas principales de dicho algoritmo son el diccionario y el corpus.

```
import gensim.corpora as corpora
dictionary = corpora.Dictionary(preprocessed_data)
corpus = [dictionary.doc2bow(text) for text in preprocessed_data]
```

Figura 9. Creación del corpus y diccionario.

El diccionario está compuesto por todas las palabras que se tendrán en cuenta para tratar nuevos textos en la ejecución del modelo. No es más que un arreglo asociativo de palabras del texto, sin repeticiones, y una clave única o Id. El propio Gensim crea dicho identificador para cada palabra en el documento.

<class 'dict'>: 'manana': 0, 'muere': 1, 'copiame': 2, 'pesao': 3, 'copiar': 4, 'instagram': 5, 'pincha': 6, '7382': 7, '9204': 8, 'envio': 9, 'choco': 10, 'pedirle': 11, 'stoy': 12, 'wifi': 13, '0321': 14, '9327': 15, 'bandec': 16, 'cerbero': 17, 'negro': 18, 'sirve': 19, 'color': 20, 'encontre': 21, 'eliminar': 22, 'perfiles': 23, 'repent': 24, 'solita': 25, 'entro': 26, 'sigo': 27, 'sigueme': 28, 'camila': 29, 'casa': 30, 'piro': 31, 'pasa': 32, 'pena': 33, 'problema': 34, 'repito': 35, 'sientas': 36, 'vida': 37, 'play': 38, 'chama': 39, 'maja': 40, 'bola': 41, 'coreo': 42, 'funciona': 43, 'mandarte': 44, 'novia': 45, 'quererme': 46, 'locuras': 47, 'feliz': 48, 'leagste': 49, 'vivir': 50, 'mala': 51, 'ensenar': 52, 'ibas': 53, 'pense': 54, 'responder': 55, 'conclusion': 56, 'simple': 57, 'pensabas': 58, 'rechaza': 59, 'malo': 60, 'pensamos': 61, 'trist': 62, 'corecta': 63, 'rechazan': 64, 'tomar': 65, 'ref': 66, 'preocupes': 67, 'hice': 68, 'papa': 69, 'prueban': 70, 'relajado': 71, 'face': 72, 'salir': 73, 'tube': 74, 'psiphon': 75, 'cuckold': 76, '192.168.173.50': 77, 'mandar': 78, 'lega': 79, 'claudia': 80, 'yesik': 81, 'enviast': 82, 'mija': 83, 'cuidt': 84, 'monte': 85, 'mete': 86, 'uiero': 87, 'eganas': 88, 'hora': 89, 'legar': 90, 'psue': 91, 'ropa': 92, 'tenia': 93, 'poquito': 94, 'tenerte': 95, 'tufaste': 96, 'wuenos': 97, 'quiero': 98, 'conjunto': 99...

Figura 10. Visualización del diccionario creado.

El corpus en Gensim es representado como una lista de relaciones $(Id, Frec)$, donde Id es el identificador de cada palabra y $Frec$ es la frecuencia de aparición de dicha palabra en su documento.

<class 'list'>: [[(0, 1)], [(1, 1)], [(2, 1)], [(3, 1)], [(4, 1)], [(5, 1)], [(6, 1)], [(7, 1), (8, 1)], [(9, 1)], [(10, 1), (11, 1), (12, 1), (13, 1)], [(8, 1), (14, 1), (15, 1), (16, 1)], [(17, 1)], [(18, 1), (19, 1)], [(20, 1)], [(21, 1)], [(22, 1), (23, 1), (24, 1)], [(25, 1)], [(26, 1), (27, 1), (28, 1)], [(29, 1), (30, 1)], [(31, 1)], [(32, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1)], [(38, 1)], [(39, 1)], [(40, 1)], [(41, 1)], [(42, 1), (43, 1), (44, 1)], [(30, 1), (45, 1)], [(46, 1)], [(47, 1)], [(48, 1)], [(49, 1), (50, 1)], [(51, 1)], [(52, 1), (53, 1)], [(54, 1), (55, 1)], [(56, 1)], [(57, 1)], [(58, 1), (59, 1)], [(32, 1), (60, 1), (61, 1), (62, 1)], [(63, 1), (64, 1), (65, 1)], [(66, 1)], [(67, 1)], [(68, 1)], [(69, 2)], [(70, 1)], [(71, 1)], [(72, 1)], [(73, 1), (74, 1)], [(75, 1)], [(76, 1)], [(40, 1)], [(77, 1)], [(78, 1)], [(13, 1)], [(79, 1)], [(80, 1), (81, 1)], [(82, 1), (83, 1)], [(84, 1)], [(85, 1)], [(86, 1)], [(87, 1)], [(88, 1), (89, 1), (90, 1), (91, 1), (92, 1), (93, 1)], [(94, 1), (95, 1)], [(96, 1)], [(38, 1), (97, 1)], [(98, 1)], [(99, 1), (100, 1), (101, 1), (102, 1), (103, 1), (104, 1), (105, 1)], [(40, 1), (106, 1)], [(107, 1)], [(6, 1), (108, 1), (109, 1)], [(110, 1)], [(111, 1)], [(112, 1)], [(113, 1)], [(78, 1), (114, 1), (115, 1), (116, 1)], [(117, 1), (118, 1)], [(119, 1)], [(75, 1), (120, 1), (121, 1), (122, 1)], [(123, 1)], [(124, 1)], [(123, 1), (125, 1)], [(121, 1)], [(126, 1)], [(44, 1), (127, 1)], [(0, 1), (69, 1), (128, 1)], [(129, 1), (130, 1)], [(131, 1)], [(132, 1)], [(133, 1)], [(134, 1)], [(135, 1)], [(0, 1), (136, 1)], [(98, 1)], [(137, 1), (138, 1)], [(139, 1)], [(140, 1), (141, 1)], [(142, 1)], [(143, 1), (144, 1), (145, 1)], [(146, 1), (147, 1)], [(148, 1)], [(149, 1...

Figura 11. Visualización del corpus.

Por ejemplo, en la figura 2.11, (0, 1) implica que la palabra cuyo identificador es cero aparece una vez en el primer documento. Del mismo modo, la palabra de identificador uno aparece también una única vez y así sucesivamente.

El método que implementa el modelo LDA de la librería Gensim necesita definir, además del diccionario y el corpus, ciertos parámetros que determinan cómo entrenar y generar el modelo de tópicos, algunos de estos parámetros son:

- iterations: número máximo de iteraciones a través del corpus al inferir la distribución temática de un corpus.
- numTopics: número de temas latentes solicitados que se extraerán del corpus de entrenamiento.
- alpha y beta: hiperparámetros que afectan la densidad de tópicos. El valor por defecto de ambos es 1/numTopics.
- chuncksize: controla cuántos documentos se procesan a la vez en el algoritmo de entrenamiento. Aumentar el tamaño del fragmento acelerará el entrenamiento.
- updateEvery: indica cada cuanto se actualizan los parámetros del modelo.
- passes: controla la frecuencia con la que entrenamos el modelo en todo el corpus. El valor por defecto es 10.

LDA es un algoritmo iterativo, un mayor número de iteraciones mejora la convergencia del modelo. Para la elección de este parámetro se atiende a una relación de compromiso entre el coste computacional, que aumenta con el número de iteraciones, y la convergencia del modelo. El valor que se elije son 300 iteraciones.

Para definir los hiperparámetros alpha y beta se fijó el número de tópicos para analizar el efecto de cada uno de ellos en el modelo generado para el corpus evaluado. Se comienza con los valores por defecto para ir aumentando progresivamente los valores de alpha y beta.

El modelo LDA requiere que se le defina también la cantidad de tópicos (k) a buscar por lo cual es importante hacer un análisis de este parámetro, previo a ejecutar el modelo. Cuando se analizan grandes volúmenes de datos para detectar los tópicos a los que hacen referencia dichos datos es común que salga a flote la siguiente interrogante: ¿cuántos tópicos estamos intentado encontrar?

Para darle solución a esta interrogante es necesario modificar la composición del modelo en un esfuerzo por encontrar la mejor combinación de parámetros, en este caso el parámetro k, para manejar el problema.

Para optimizar el valor de k se decidió generar diferentes modelos LDA con valores de k distintos y luego de esos modelos escoger el modelo óptimo teniendo en cuenta el valor de coherencia de cada uno.

El caso de medir la coherencia de los tópicos se ha estudiado recientemente para remediar el problema de que los modelos de tópicos no garantizan la interpretabilidad de sus resultados Röder et al. (2015).

Medir la coherencia de tópicos es una buena forma de comparar diferentes modelos de tópicos en función de su interpretabilidad humana KUMAR (2017). El valor de coherencia c_v captura el número óptimo de tópicos al otorgar a la interpretabilidad de estos una puntuación de coherencia. Para calcular dichos valores se utlizó el paquete CoherenceModel de la librería Gensim.
El procedimiento a seguir en el método propuesto puede ser observado en la. figura 12.

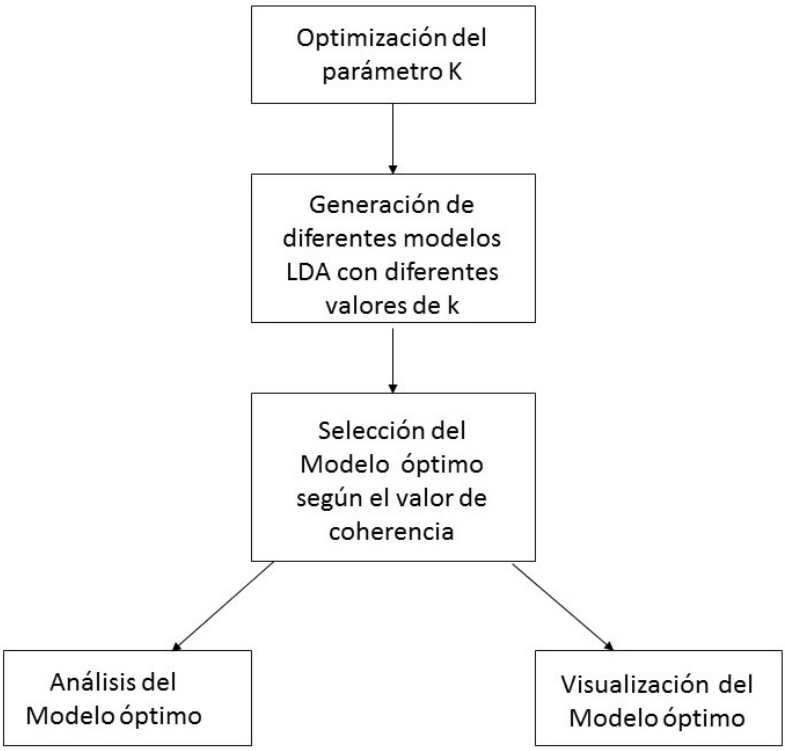

Figura 12. Optimización del parámetro k. Fuente: elaboración propia.

A continuación se muestra el código utilizado para calcular los valores de coherencia para cada uno de los modelos LDA generados.

Los objetos de esta clase CoherenceModel permiten construir y mantener un modelo de coherencia de

```
import gensim
from gensim.models.coherencemodel import CoherenceModel
def compute_coherence_values(dictionary, corpus, texts, limit, start
    =2, step=1):
     coherence_values = []
     model_list = []
     for num_topics in range(start, limit, step):
         if __name__ == '__main__':
             freeze_support()
             model = gensim.models.ldamodel.LdaModel(corpus,
                                                     num_topics=
                                                        num_topics,
                                                     id2word=dictionary
                                                        ,
                                                     passes=value,
                                                     random_state=value
                                                        )
             model_list.append(model)
             coherencemodel = CoherenceModel(model=model,
                                             texts=texts,
                                             dictionary=dictionary,
                                             coherence='c_v')
             coherence_values.append(coherencemodel.get_coherence())
     return model_list, coherence_values
```

Figura 13. Función utilizada para calcular los valores de coherencia.

tópicos. Los variables que recibe como parámetro son: *model, texts, dictionary* del modelo lda creado anteriormente y se especifica en *coherence* la medida de coherencia a utilizar *c_v* a la cual se le debe proporcionar el texto Kapadia (2019).

Una vez determinado el modelo óptimo dado el valor de coherencia obtenido se procede a generar el modelo LDA.

El modelo LDA utilizado en esta herramienta fue optimizado para los datos de prueba procesados, cuando el *dataset* cambie será importante optimizar los parámetros nuevamente para mantener la calidad de los resultados.

2.5. Detección e interpretación de tópicos

2.5.1. Visualización

El objetivo de presentar una visualización de tópicos, es representar los datos a la salida del modelo de forma interpretable. Como establecen Chaney and Blei (2012) los objetivos de la visualización deben ser:

```
import gensim
def generateOptimalModel(optimal_numTopics):
    lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=
                                              optimal_numTopics,
                                           random_state=value,
                                           chunksize=value,
                                           passes=value,
                                           iterations=value,
                                           alpha=value)
    return lda_model
```

Figura 14. Generación del modelo óptimo LDA.

- Resumir el corpus visualizando la composición de los tópicos.
- Revelar las relaciones entre los documentos evaluados y los tópicos descubiertos
- Revelar relaciones entre los documentos evaluados.

Estos objetivos establecen las bases de la extracción de información de una colección de documentos empleando un modelo de tópicos.

En la literatura que versa sobre visualización de tópicos se suele hacer especial hincapié en el primero de los objetivos presentados. Es decir, las visualizaciones suelen centrarse en la de la composición de los tópicos descubiertos. Este tipo de visualizaciones permiten comprender el corpus de manera global, pero no prestan atención a los documentos a nivel individual. Por tanto, se desatienden las relaciones que se establecen entre documentos Vázquez Marcos (2017).

Sievert and Shirley (2014) proponen un método –LDAvis- para la visualización de la composición de los tópicos. En este trabajo se emplea este método de visualización, se introduce a continuación sus bases de funcionamiento de acuerdo a lo expuesto por Sievert and Shirley (2014).

La herramienta LDAvis permite una visualización interactiva que ofrece las siguientes posibilidades:

- Tópicos representados mediante círculos en el plano de dos dimensiones, donde el área de cada circulo representa la predominancia de cada tópico en el corpus. Los tópicos son ordenado en orden decreciente respecto a su predominancia.
- La posición de cada tópico se computa mediante la distancia entre tópicos, usando MDS para proyectar estas distancias al plano de dos dimensiones.
- Se representa mediante diagramas de barras las palabras más relevantes para el tópico seleccionado. Se superponen una barra roja y otra azul para cada término que representan la frecuencia del término de la palabra evaluado en todo el corpus, es decir, el número de veces que se repite esa palabra en el conjunto de documentos evaluados y la frecuencia del término estimada de la palabra en el tópico seleccionado respectivamente.

Para poder graficar fácilmente los tópicos obtenidos del modelo aplicado se utilizó la librería de Python

pyLDAvis

```
import pyLDAvis.gensim
def show(optimal_model, corpus, dictionary):
    lda_display = pyLDAvis.gensim.prepare(optimal_model, corpus,
       dictionary, sort_topics=True)
    pyLDAvis.display(lda_display)
    pyLDAvis.save_html(lda_display, 'lda.html')
    pyLDAvis.show(lda_display)
```

Figura 15. Función utilizada para visualizar los tópicos obtenidos.

3. Resultados y discusión

En el presente capítulo se realiza la validación del método desarrollado mediante varios criterios para determinar que la solución propuesta responde satisfactoriamente a los objetivos que se trazaron. Se realizan varias pruebas al código para asegurar la calidad del método propuesto verificando que no existan fallas.

3.1. Optimización de hiperparámetros α y β

En la primera fase de la generación del modelo LDA, se estudia el efecto de los parámetros α y β sobre los resultados del modelo. Tras este proceso se decide establecer los siguientes valores: α=0.6 β=0.01

El proceso que condujo a la selección de estos valores se explica a continuación.

En primer lugar, con un número de 5 tópicos preestablecido, se estudió la salida del modelo LDA para el conjunto de evaluación con los valores por defecto de $\alpha(\alpha = 0{,}1)$ y $\beta(\beta = 0{,}01)$. Se observó que a la salida del modelo la mayoría de los documento eran asignados a los distintos tópicos con probabilidades superiores a 0.9. Esto quiere decir, que cada documento expresaba información de pertenencia a un único tópico.

Esta representación no era válida para las colecciones bajo estudio. Dada la heterogeneidad en cuanto a su contenido, una categorización de los documentos tan rígida no expresaba relaciones entre documentos. Como se explica en el segundo capítulo, el valor de α, influye en como se asigna la pertenencia de un documento a los distintos tópicos. A valores más grandes, se considera que un documento está representando por la combinación de un mayor número de tópicos. De esta manera, si una colección de documentos no está claramente categorizada se recomienda el uso de valores de α cercanos a 1.

Ante pruebas con valores de α del orden de 0.8 y 0.9 la salida del modelo para los dos conjuntos de evaluación presentaba documentos en los que casi ningún documento pertenecía a un tópico concreto con probabilidad mayor que 0.5. Esto implica, que los documentos quedaban representados como una combinación de tópicos con parecido valor de pertenencia para cada uno.

Lo que se pretende es tener una representación en un punto medio de los dos extremos antes expuestos. Es decir, tener documentos representados como una combinación de tópicos, pero en los que uno de los tópicos tenga una importancia significativa.

Sobre el parámetro β, se deja su valor por defecto. Cabe mencionar que se probó el efecto de este parámetro sobre la representación. Como se explica en el segundo capítulo, valores más grande de β, aumenta el número de palabras que definen un tópico. A priori se podría pensar que sería necesario aumentar el valor de este parámetro, pero aumentar este parámetro se traduce en que los tópicos estuvieran formados por conjuntos de palabras relevantes muy similares. Esto se observó en la composición de los tópicos mediante PyLDAvis. Por este motivo se selecciono su valor por defecto

3.2. Óptimización del número de tópicos (k)

Como se mostró en le capítulo anterior la solución es un método que utiliza el modelo LDA. Éste modelo impone definir de antemano la cantidad de tópicos a encontrar sobre el set de datos a analizar. Es por esto que el conocimiento del valor de dicho parámetro es de suma importancia conocer antes de entrenar el modelo y por lo tanto comenzaremos explicando cómo optimizar dicho parámetro. Posteriormente, utilizando el parámetro óptimo, se genera el modelo en cuestión observando los tópicos obtenidos y finalmente analizaremos dichos tópicos mediante visualizaciones.

La forma utilizada para obtener el número óptimo de tópicos es construir varios modelos LDA con diferente número de tópicos (k) y tomar aquel que devuelva el valor de coherencia (c_v) [1] más grande considerando el contexto. Dicho valor de coherencia es una forma simple de ver qué tan bueno es el modelo. Para la elección del valor k, el c_v no tiene que ser simplemente el más grande y es por eso que también es importante la cantidad de datos que uno está procesando. Si observamos las mismas palabras clave repetidas en muchos de los tópicos, es probablemente una señal de que el parámetro k es muy grande. El siguiente gráfico (fig 16) representa los diferentes valores de coherencia obtenidos para distintos k con los que se entrenó el modelo LDA. Además, hay líneas que conectan el valor medio de cada valor de coherencia para los valores de k. Ponweiser (2012)

El gráfico se muestra a partir de los siguientes valores:

Num Topics = 3 has Coherence Value of 0.6655

Num Topics = 4 has Coherence Value of 0.6788

Num Topics = 5 has Coherence Value of 0.6790

Num Topics = 6 has Coherence Value of 0.6873

Num Topics = 7 has Coherence Value of 0.695

[1] https://rare-technologies.com/what-is-topic-coherence/

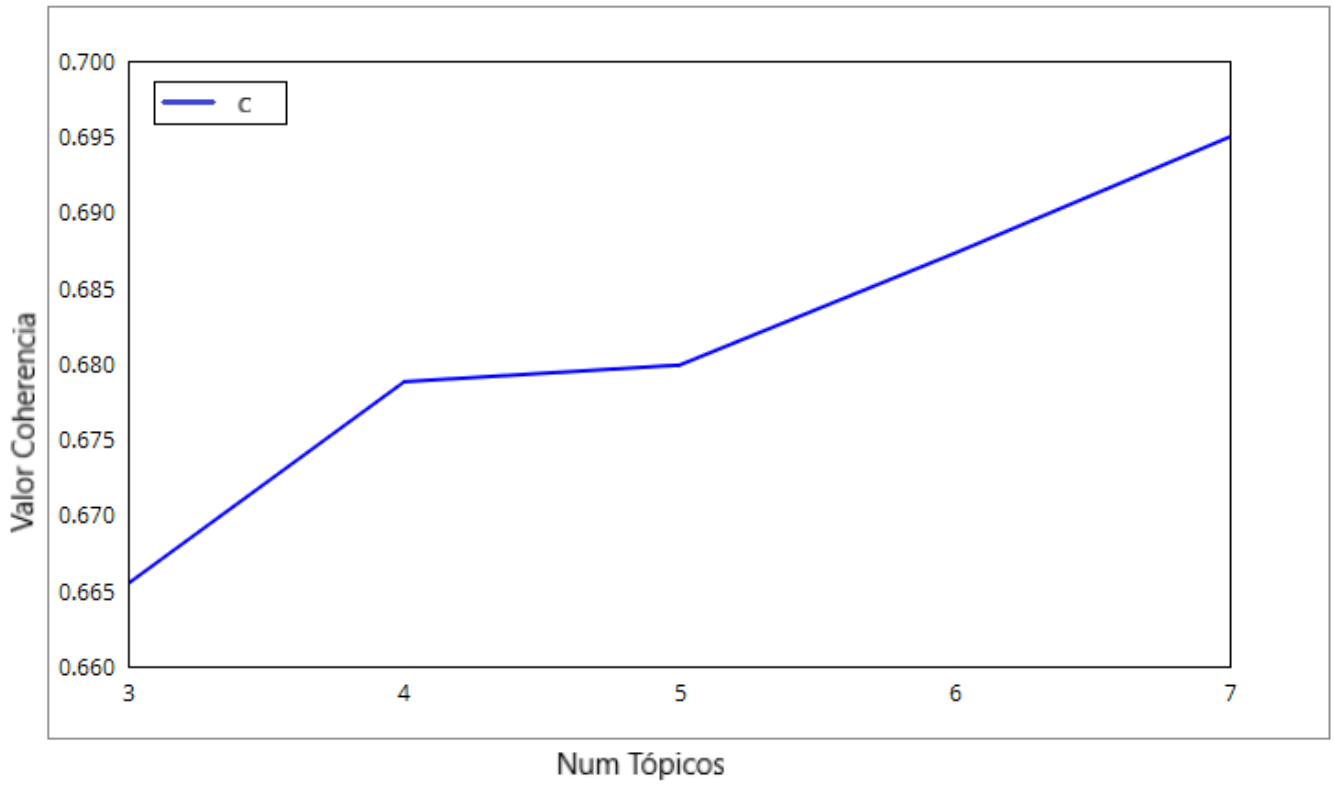

Figura 16. Valores de coherencia para diferentes números de tópicos.

Dados los hiperparámetros fijos y una variedad de temas K que van desde de 3 a 7 temas, vemos que un número de 4 tópicos representa lo mejor para el corpus.

Cuando el valor de coherencia sigue aumentando, lo mejor es elegir el que dé el máximo valor de coherencia antes de que la curva se empiece a hacer más llana. Para ilustrar a continuación mostramos algunos valores de tiempo que tomó generar cada uno de los modelos y el proceso total de cálculo del valor de coherencia:

— 76.1792426109314 seconds for model with k = 3 —

— 72.97278571128845 seconds for model with k = 4 —

— 68.66336512565613 seconds for model with k = 5 —

— 67.05760836601257 seconds for model with k = 6 —

— 66.08022141456604 seconds for model with k = 7 —

3.3. Observaciones

Como hemos mencionado previamente, el modelo LDA fue construido, donde cada tópico es una combinación de palabras claves y cada palabra clave contribuye con un cierto peso al tópico.

Podemos ver cada palabra clave de cada tópico y el peso o importancia de cada palabra clave como se muestra a continuación

```
[(0, '0.027*"mira" + 0.015*"paso" + 0.015*"coreo" + 0.012*"hora" + 0.010*"mando"'),
(1, '0.016*"casa" + 0.011*"tenia" + 0.011*"madre" + 0.010*"mande" + 0.006*"necesito"'),
(2, '0.039*"studio" + 0.015*"vale" + 0.011*"pasa" + 0.009*"tesis" + 0.009*"vida"'),
(3, '0.023*"cosa" + 0.015*"falta" + 0.013*"puedes" + 0.013*"quieres" + 0.012*"quiero"')]
```

Figura 17. Palabras claves por tópicos.

Los pesos reflejan qué importancia otorga cada palabra clave a ese tema.

Observando las palabras claves se debe poder resumir cuál es el tópico en cuestión al que pertenecen. En el caso del tópico cero podríamos decir que está asociado al tópico “Envío de correo” . De la misma forma, podemos ir determinando las palabras claves de los otros tópicos para definir a qué hace referencia cada uno.

3.4. Visualización

A continuación se presentan tres figuras que ilustran la apariencia que presenta la herramienta LDAvis.

Cada burbuja en la parte izquierda de la visualización representa un tópico. Cuanto más grande la burbuja, más predominante es ese tópico. Un buen modelo de tópicos es aquel que tiene burbujas bastante grandes y que no se solapan, dispersas en todo gráfico en lugar de estar todas juntas y clusterizadas en un único cuadrante.

Un modelo con muchos tópicos seguramente tendrá burbujas pequeñas, ubicadas en una misma región del gráfico y con muchos casos de solapamiento. Si movemos el cursor sobre cada una de las burbujas, las palabras y las barras del gráfico de la derecha se actualizarán. Estas palabras son las palabras clave más importantes del tópico seleccionado.

El diagrama de barras de la figura 18, en la que no se ha seleccionado ningún tópico, representa las 30 palabras con mayor prominencia. Por otra parte la figura 20 ilustra que si seleccionamos una palabra concreta, podemos ver gráficamente a que tópicos pertenece. El área de los círculos, pasa a indicar el grado de pertenencia de la palabra al tópico.

Mientras que las barras azules representan al atributo *Silency*, una medida de cuánto un término dice acerca de un tópico; y las barras en rojas al atributo *Relevance*, un peso promedio de la probabilidad de

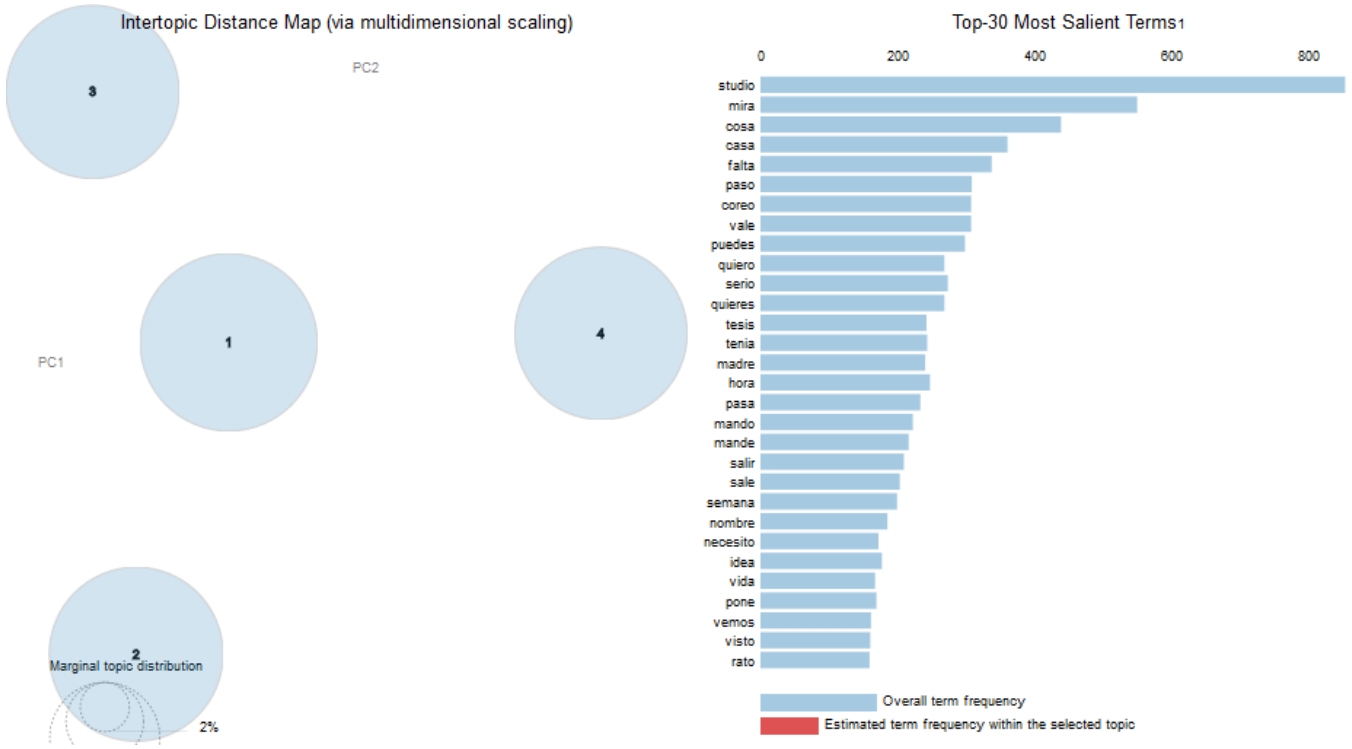

Figura 18. Visualización de los tópicos del modelo generado.

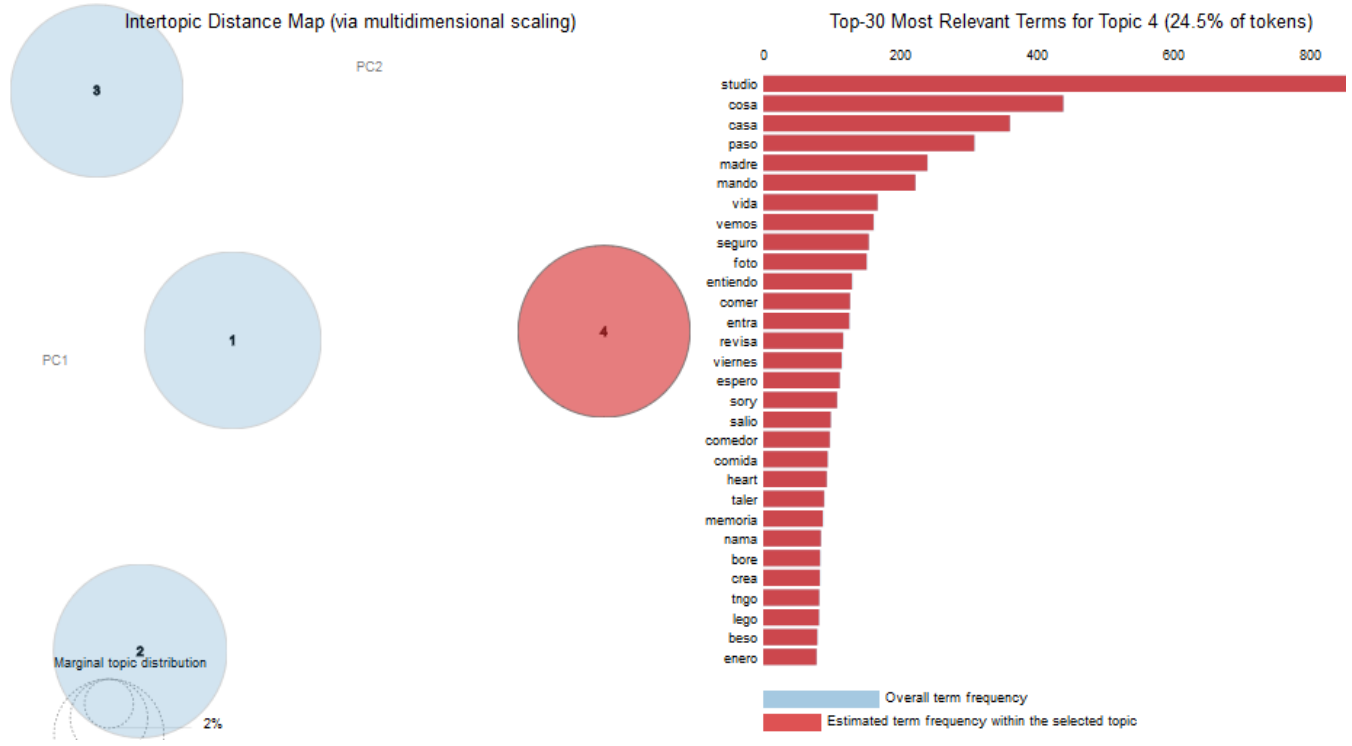

Figura 19. Visualización de la composición de un tópico.

una palabra dado un tópico y de esa palabra dado el tópico normalizado por la probabilidad del tópico.

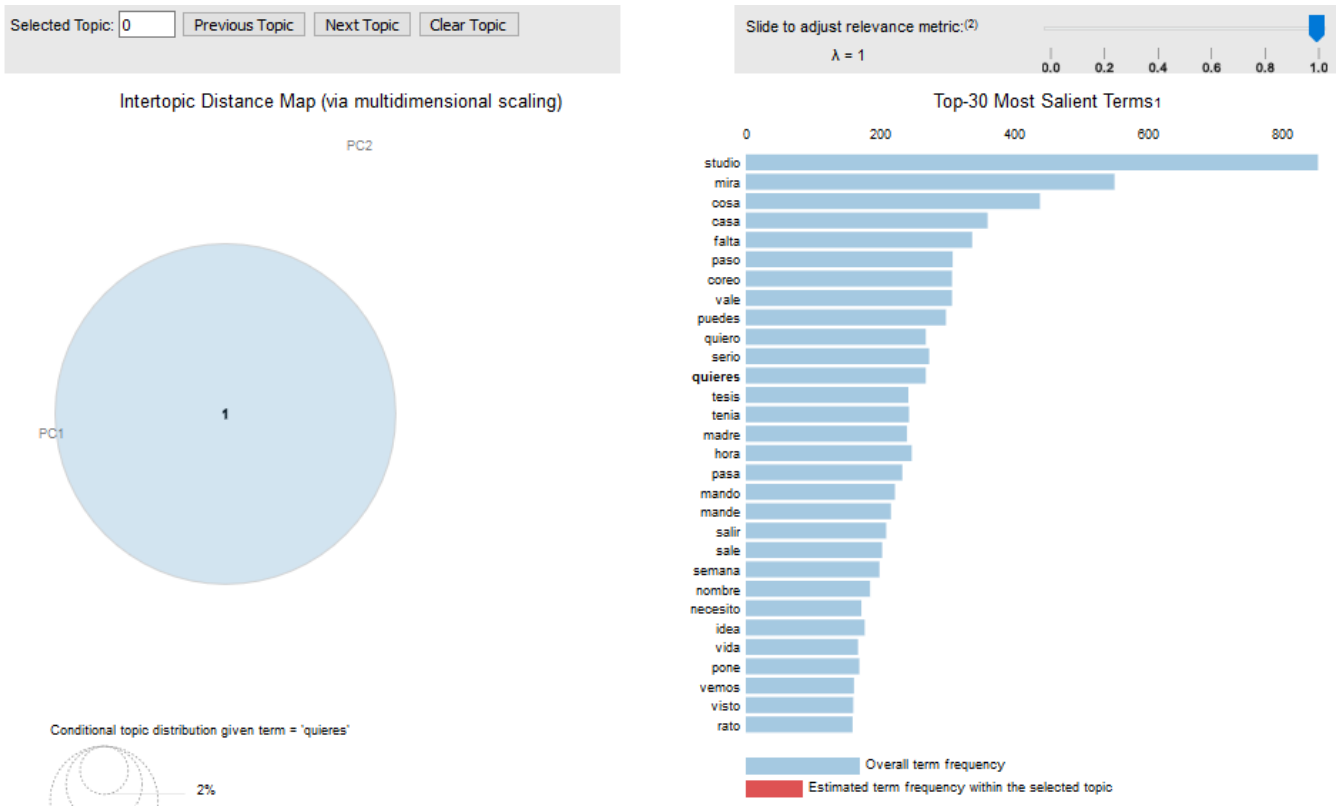

Figura 20. Pertenencia de una palabra al tópico.

Por último cabe mencionar, que como se observa en la parte superior izquierda, en la herramienta de visualización LDAvis hay un parámetro libre $\lambda \in (0,1)$. Este parámetro influye a la hora de ordenar los términos más relevantes de cada tópico.

4. Conclusiones

Al finalizar la presente investigación, arribamos a las siguientes conclusiones que dan solución al objetivo general de la misma:

- La detección de tópicos permitió determinar los temas de conversación, así como palabras claves relacionadas a cada tema dentro de las interacciones de los usuarios de la red de mensajería instantánea de la UCI.
- Se obtuvo un método implementado en Python que facilitó: el procesamiento de un gran volumen de documentos de manera rápida y simple, y el poder detectar tópicos o temas a los que hacen referencia los usuarios de una red social con visualizaciones que permiten analizar de manera sencilla los tópicos.
- Al utilizar el modelo LDA se puede observar que la determinación de palabras claves dentro de un texto es fundamental para el buen desempeño del modelo LDA, donde el pre procesamiento del texto juega un rol fundamental en la segmentación, normalización y eliminación del ruido.
- La validación realizada a los parámetros α, β, y k del modelo LDA, permitieron obtener un modelo óptimo para el set de datos procesados.

Referencias

Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.

Andrade Alvarado, V. d. R. et al. (2015). *Clasificación de artículos científicos*. PhD thesis, Universidad de Concepción. Facultad de Ingeniería. Departamento de ....

Brun, R. E. and Senso, J. A. (2004). Artículo minería textual. *El profesional de la información*, 13(1):11.

Chaney, A. J.-B. and Blei, D. M. (2012). Visualizing topic models. In *Sixth international AAAI conference on weblogs and social media*.

Kapadia, S. (2019). Evaluate Topic Models: Latent Dirichlet Allocation (LDA). *Medium*.

KUMAR, R. (2017). Natural language processing. In *Machine Learning and Cognition in Enterprises: Business Intelligence Transformed*, pages 65–73. Apress, Berkeley, CA.

Ng, W. (2016). *NEW DIGITAL TECHNOLOGY IN EDUCATION*. Springer.

Ponweiser, M. (2012). *Latent Dirichlet allocation in R*. PhD thesis, WU Vienna University of Economics and Business.

Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408.

Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70.

Vázquez Marcos, J. (2017). Modelado de tópicos para perfilado de blogs. B.S. thesis.

Vilares Calvo, D. (2014). Análisis de contenidos en twitter: clasificación de mensajes e identificación dela tendencia política de los usuarios. B.S. thesis, Universidad de Coruña.