Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Programa de Maestría en Computación

Evaluación del uso de distintas métricas de distancia de texto en un algoritmo agregado para la imputación de valores faltantes mediante clasificación

(Evaluation of using different text distance metrics in an ensemble algorithm for missing values imputation through classification)

Tesis para optar al grado de Magister Scientiae en Computación

Autor:
José Andrés Mena Arias

Profesor Asesor:
Luis Alexánder Calvo Valverde

Junio, 2017

Resumen

Actualmente, muchas bases de datos alrededor del mundo sufren un problema general de valores faltantes, el cual es causado por diversas razones que van desde fallos de *hardware* hasta valores no ingresados por los usuarios. La idea de imputación de datos consiste en utilizar algún método que permita encontrar valores plausibles para aquellos atributos en donde el valor no existe. Cuando el valor faltante puede inferirse a partir de un valor de tipo texto, el problema se convierte en uno de algoritmos de clasificación donde documentos de texto son organizados en diferentes categorías (valores faltantes a imputar). Esto involucra determinar computacionalmente qué tan similar es un texto a otro. La literatura existente sobre métodos para resolver este problema es muy extensa, sin embargo, los métodos estadísticos (que utilizan métricas de similitud sobre vectores de palabras) han mostrado resultados muy favorables en los últimos 25 años en el campo de la minería de texto [38]. Adicionalmente, el modelado de temas ha surgido como una prometedora alternativa a los métodos existentes, al lograr reducción dimensional e incorporación de la semántica en la clasificación de documentos [30]. Este proyecto se enfoca en la evaluación de técnicas de representación y medidas de similitud de texto tradicionales (vectores de palabras, Coseno y Jaccard), con respecto a métodos que involucran modelado de temas y comparación de distribuciones de probablidad (Asignación de Dirichlet Latente y Divergencia Kullback-Leibler). Para ello, se analizan estadísticamente los resultados de distintos experimentos que involucran la utilización de las métricas ya mencionadas, de forma individual y combinadas, para clasificar conjuntos de datos de documentos de texto.

En términos generales, los resultados obtenidos muestran que los modelos de representación de datos mediante Asignación de Dirichlet Latente, combinados con la métrica de entropía relativa, obtienen valores de exactitud estadísticamente comparables a los obtenidos utilizando las técnicas tradicionales. El modelado de temas logra abstraer miles de palabras en menos de 60 temas en el caso de los experimentos principales. Los resultados también evidencian desventajas, áreas de mejora y escenarios potenciales en los cuales dichos modelos podrían lograr un mejor desempeño.

Abstract

Nowadays, there is a general problem of missing values in databases around the world, which is caused by several reasons going from hardware malfunctions to non-mandatory fields in forms. Data imputation can be defined as the use of some method to find plausible values for those missing. When the missing value can be inferred from a text value attribute, then the problem can be seen as a classification algorithms problem where text documents should be organized within categories representing the plausible missing values. It also implies the problem of calculating how similar is a text value with respect to another. Existing literature about solving this kind of problems is extensive, however, during the last 25 years the statistical methods (where similarity functions are applied over vectors of words) have achieved good results in many areas of text mining [38]. Additionally, topic modeling has arisen in the last years as a promising alternative to existing methods by achieving dimensional reduction and incorporating the semantic factor when classifying documents [30]. This project is focused on the evaluation of traditional data representation techniques and similarity metrics (words vectors, Cosine and Jaccard) respect to topic modeling techniques and probability distributions comparison (Latent Dirichlet Allocation and Kullback-Leibler Divergence). An statistical analysis is applied to the results obtained after running several experiments that involved the mentioned metrics, both individually and combined, to classify data sets of text documents.

At a high level, the results show that the accuracy scores achieved by using document representations obtained thought Latent Dirichlet Allocation, combined with the relative entropy metric, were statically similar to the ones obtained by using traditional text classification techniques. The topics modeling manages to abstract thousands of words in less than 60 topics for the main set of experiments. The results also highlight cons, improvement areas and potential scenarios where such models could achieve a better performance.

Agradecimientos

Agradecimiento especial al profesor Luis Alexander Calvo por toda la ayuda y retroalimentación brindada a lo largo del tiempo que se requirió para llevar a cabo la investigación. De igual forma, se agradece al programa de Maestría en Computación del Instituto Tecnológico de Costa Rica por facilitar el proceso académico mediante su convenio con la empresa Intel Costa Rica.

Programa de Maestría
en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Evaluación del uso de distintas métricas de distancia de texto en un algoritmo agregado para la imputación de valores faltantes mediante clasificación"

TRIBUNAL EXAMINADOR

Máster Luis Alexander Calvo Valverde
Profesor Asesor

Doctor Carlos González Alvarado
Profesor Lector

Máster Jacqueline Tatiana Solís Céspedes
Profesor Externo

Doctor Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

Junio, 2017

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Siglas

KL Kullback-Leibler. 19

KNN Vecinos más cercanos (del inglés K Nearest Neighbors). 18

LDA Asignación de Dirichlet Latente (del inglés Latent Dirichlet Allocation). 18

Glosario

Asignación de Dirichlet Latente Modelo probabilístico utilizado para clasificar documentos en temas, tomando en cuenta que un mismo documento puede tener varios temas latentes y cada tema puede ser representado como una distribución de palabras. 27

bagging Método agregado que consiste en entrenar un algoritmo utilizando diferentes muestras aleatorias, elegidas del conjunto de datos de entrenamiento original con reemplazo. 26

distancia de Coseno Métrica del grado de similitud entre dos documentos utilizando el coseno del ángulo formado por las respectivas representaciones vectoriales. 24

divergencia Kullback-Leibler También conocida como la entropía relativa, sirve para medir las diferencias entre dos distribuciones de probabilidad y es utilizada en *clustering* de documentos. 24

exactitud Porcentaje de aciertos en un experimento tomando en cuenta el total de la población. 39

F1 macro ponderado Calcula un promedio ponderado de la precisión y de la sensibilidad obtenidas para todas las clases involucradas en el proceso de clasificación. 39

Jaccard Tamaño de la intersección de las representaciones vectoriales de dos documentos de texto, dividido entre el tamaño de la unión de los mismos conjuntos. 24

precisión Proporción de instancias de cierta clase correctamente clasificadas con respecto al total de instancias clasificadas bajo esa clase. 39

semántica Se refiere a los aspectos del significado, sentido o interpretación de signos lingüísticos como símbolos, palabras, expresiones o representaciones formales. 31

sensibilidad Proporción de instancias de cierta clase correctamente clasificados sobre el total de instancias donde el valor correcto es esa clase. 39

stemming Proceso en el cual se reduce una palabra a su forma derivada, base o raíz. 54

stop words Se refiere a la lista de palabras más comunes en un lenguaje (tales como artículos y pronombres) que generalmente no aportan mucho valor al momento de realizar comparaciones entre textos. 54

tupla Este término también es conocido como registro y se define como una función finita que mapea (asocia unívocamente) nombres con valores. 20

vector de palabras Técnica que consiste en representar cada documento de texto como un vector, donde cada entrada corresponde a un término y el valor de la entrada es la frecuencia de ese término en el documento. 35

Capítulo 1

Introducción

Actualmente, muchas bases de datos alrededor del mundo sufren un problema general de valores faltantes. Dicho problema puede ser causado por diferentes razones, tales como fallos mecánicos durante la recolección de los datos, información que no existía al momento de crear la base de datos, formularios con campos no obligatorios que generan valores nulos no ingresados por los usuarios, o fallos informáticos que ocasionan pérdida de datos [3, 40]. En términos generales, las dos posibles soluciones a este problema son: la eliminación del registro o atributo que contenga valores faltantes o la imputación de dicho valor con algoritmos de aprendizaje máquina o de minería de datos [3].

La eliminación de datos puede ocasionar un decremento en la cantidad de información valiosa disponible, además de sesgos importantes al momento de analizar esos datos [42]. Surge entonces la idea de imputación de datos, en la cual se utiliza algún método que permite encontrar valores plausibles para aquellos atributos en donde el valor no existe [42]. Una vez imputados los valores, los datos pueden ser considerados como confiables para el procesamiento o análisis y dicha confiabilidad depende de las suposiciones, algoritmos y datos de entrenamiento que se utilicen [3, 42].

La idea general de los algoritmos de imputación de datos es encontrar relaciones entre los datos completos (valores observados) y con base en éstas, aproximar los

valores a imputar en los datos incompletos (valores faltantes) [29]. De esta forma, los métodos existentes hacen uso de funciones para estimar valores faltantes a partir de valores completos, por ejemplo, un modelo polinomial de regresión que aproxime un valor faltante en una serie de valores numéricos continuos obtenidos mediante un sensor [40]. Sin embargo, hay situaciones en las cuales los valores completos no son numéricos, pero que aún existe la posibilidad de que los mismos puedan ser utilizados para estimar el valor faltante. Este proyecto se enfoca en el problema de imputación de valores faltantes donde los valores completos son de tipo texto, de forma que los métodos a utilizar deben ser capaces de aplicar funciones sobre conjuntos de caracteres o palabras, comúnmente conocidos en la literatura como documentos [38].

Se presenta el siguiente ejemplo para aclarar lo descrito: Se tiene una base de datos que almacena reportes de errores de sistemas que se reciben de forma constante todos los días. Cada registro almacena la siguiente información: nombre del sistema, descripción del error (campo de texto libre), persona que resolvió el problema, descripción de la solución (texto libre) y causa del problema (categoría que se selecciona de una lista: error en el código, error de datos, error de usabilidad, error de proceso, etc.). La causa del problema es ingresada por la persona que resolvió el error en un formulario y dicho campo no es obligatorio, mientras que el texto libre con la descripción de la solución sí lo es.

Si en algún momento se necesitara realizar un análisis de los datos para determinar tendencias en el origen de los errores, se encontrará que la base de datos presenta una gran cantidad de valores faltantes para el atributo 'causa del problema' debido a personas que no ingresaron dicho valor. Aun así, dicha categoría podría inferirse a partir de los otros campos de tipo texto libre tales como la descripción del problema y la descripción del error. Aunque esto es una tarea que podría ser realizada por personas, cuando se habla de miles de registros con valores faltantes, surge la necesidad de buscar soluciones automatizadas en las que un programa sea capaz de encontrar esas relaciones entre el texto libre y las categorías de forma similar a como lo haría

el cerebro humano.

En este contexto, el problema de los valores faltantes puede verse como un problema de clasificación, donde un algoritmo debe ser capaz de organizar los valores completos (atributos de tipo texto) en diferentes categorías. Dichas categorías representan el valor faltante a imputar. Actualmente, se han propuesto diversos algoritmos en los campos de la minería de datos y del aprendizaje máquina para resolver problemas de clasificación de forma supervisada [40, 42, 19]. Para el caso específico de clasificación de documentos, destacan aquellos algoritmos que hacen uso de medidas de similitud de texto, tal como Vecinos Más Cercanos (KNN de la traducción al inglés *K-Nearest-Neighbors*)[39].

Por lo tanto, el problema de clasificación de documentos mediante aprendizaje supervisado, implica también el problema de determinar computacionalmente qué tan similar es un texto a otro. Actualmente la literatura sobre métodos para atacar este problema es muy extensa, sin embargo, los métodos estadísticos, que intentan crear representaciones matemáticas de texto y que no toman en cuenta semántica ni propiedades lingüísticas, han mostrado resultados muy favorables en los últimos 25 años en el campo de la minería de texto [38]. Ejemplos de estos métodos estadísticos son todos aquellos que implican la representación de documentos mediante vectores de palabras, y la utilización de medidas de distancia entre vectores tales como Coseno o *Jaccard* [43, 37]. Estos métodos también son conocidos como medidas de similitud basadas en términos, que se diferencian de aquellos basados en cadenas de caracteres o *corpus* semánticos [43].

Por otro lado, el modelado de temas ha surgido como una prometedora alternativa a los métodos existentes. Éstos han logrando buenos resultados al clasificar documentos haciendo uso del concepto de similitud entre distribuciones de probabilidad, reducción dimensional e incorporación del factor semántico [5, 8, 18, 31]. La Asignación de Dirichlet Latente (LDA por sus siglas en inglés) es una de las técnicas

más populares en esta área, tanto en aprendizaje supervisado como no supervisado [30].

Dado todo lo expuesto anteriormente, este proyecto se enfoca en la evaluación de técnicas de representación y medidas de similitud de texto tradicionales (vectores de palabras, Coseno y Jaccard), con respecto a métodos que involucran modelado de temas y comparación de distribuciones de probabilidad como LDA y la Divergencia Kullback-Leibler (Divergencia KL). Para ello, se analizan estadísticamente los resultados de distintos experimentos que involucran la utilización de las métricas ya mencionadas, de forma individual y combinadas, para clasificar conjuntos de datos de documentos de texto.

El resto del documento se organiza de la siguiente manera. En el capítulo 2 se presenta el marco teórico donde se describe el problema general de imputación de valores faltantes y se definen las técnicas de representación de documentos de texto, métricas de distancia de texto, conceptos del modelado de temas supervisado y algoritmos de clasificación utilizados. El capítulo 3 resume los aspectos generales de la investigación, incluyendo el planteamiento del problema, la hipótesis de investigación y los objetivos, así como una descripción del desarrollo y la ejecución de los experimentos. Los resultados se muestran en el capítulo 4 con su respectivo análisis estadístico. Finalmente, en el capítulo 5 se detallan las conclusiones de la investigación y el trabajo futuro.

Capítulo 2

Marco teórico

En esta sección se explican los principales conceptos a estudiar en este proyecto, alrededor del problema de valores faltantes y la clasificación de documentos de texto.

2.1. El problema general de imputación de valores faltantes

Se tiene un esquema relacional denotado por $R(A_1, A_2, ..., A_n)$ donde R es el nombre del esquema, $A_1, A_2, ..., A_n$ corresponde a una lista de n atributos y $dom(A_i)$ es el dominio de cierto atributo A_i con $1 \leq i \leq n$. Se puede representar un conjunto de datos $r(R)$ como un conjunto formado por m tuplas $r = t_1, t_2, \ldots, t_m$. Cada tupla t es un vector de n dimensiones, de forma que $t = (v_1, v_2, \ldots, v_n)$ donde $v_i = t[A_i] \in dom(A_i)$ para $1 \leq i \leq n$ [12].

Se define entonces el problema de los valores faltantes como la existencia de un subconjunto X de r, formado por k tuplas $x_1, x_2, ..., x_k$ donde para todo i con $1 \leq i \leq k$ existe un j tal que $x_i[A_j] = x_{i,j} = \beta$ siendo β un valor faltante que no pertenece al dominio del atributo A_j y con $1 \leq j \leq n$. De esta forma, la idea de imputación de datos faltantes consiste en encontrar una función f que al ser aplicada sobre el valor de un atributo A_l en la tupla i se obtiene un valor plausible para $x_{i,j}$, con $1 \leq l \leq n$,

$l \neq j$, $1 \leq i \leq k$ y $\beta \neq x_{il} \in dom(A_l)$.

$$f(x_{il}) = x'_{ij} \tag{2.1}$$

En 2.1 se presenta de forma simple el problema de imputación de valores faltantes para una tupla i donde $x_{il} \in dom(A_l)$ y $x'_{ij} \in dom(A_j)$. x'_{ij} representa un valor plausible para $x_{ij} = \beta$. Como ya se ha mencionado, el problema de los valores faltantes puede verse como un problema de clasificación, en el que un algoritmo debe ser capaz de organizar los valores completos en diferentes categorías que representan los valores plausibles a imputar. Este tipo de algoritmos se describen en la siguiente sección.

2.2. Algoritmo de clasificación KNN

Siguiendo con la notación descrita en la sección anterior, si A_i es el atributo en el cual existen valores faltantes en el conjunto de datos, entonces un algoritmo de clasificación esperaría que el tipo de datos de A_i sea categórico, de forma que al aplicar una función de clasificación sobre uno o varios de los otros atributos se obtenga como resultado una categoría c tal que $c \in dom(A_i)$.

En este proyecto se estudia el algoritmo KNN por ser éste un clasificador con aprendizaje supervisado y que puede hacer uso de medidas de similitud de texto que se explicarán más adelante. KNN es un algoritmo de clasificación que selecciona las k observaciones más cercanas a un determinado valor de acuerdo a alguna métrica de distancia. Aunque es categorizado como algoritmo de aprendizaje máquina, KNN es considerado un aprendiz perezoso ya que no se realiza un aprendizaje propiamente dicho sino que más bien la clasificación se realiza utilizando el mismo conjunto de entrenamiento [29].

En el algoritmo 1 se muestra la idea general de KNN utilizado para imputación de valores faltantes. En este caso, el algoritmo recibe un conjunto de datos de entre-

namiento E y un conjunto de datos con valores faltantes X, ambos bajo el mismo esquema relacional de n atributos donde el i-ésimo atributo en E es de tipo categórico, mientras que en X todos los valores para ese atributo son valores faltantes β. La función $EncontrarVecinos$ retorna una lista de k tuplas tomadas del conjunto E donde los valores de los atributos diferentes a i son más cercanos a los valores de los mismos atributos en x con respecto a alguna función de distancia. En la función $ObtenerCategoriaPorVotacion$ se obtiene el valor con más ocurrencias en el atributo i de las tuplas almacenadas en *vecinos*. Dicha categoría finalmente se reemplaza por el valor β en $x[i]$ (Se imputa el valor faltante).

Algoritmo 1: Algoritmo KNN de clasificación para imputación valores faltantes

Input: k;E;X;i
Sea E un conjunto de tuplas completas donde el i-ésimo atributo es categórico (conjunto de entrenamiento);
Sea X un conjunto de tuplas con valores faltantes en el i-ésimo atributo;
foreach *tupla x en X* **do**
    $vecinos \leftarrow$ EncontrarVecinos(k, x, E, i);
    $categoria \leftarrow$ ObtenerCategoriaPorVotacion$(vecinos, i)$;
    $X[i] \leftarrow categoria$;

Se puede observar que KNN involucra la comparación de valores que pertenecen a un registro con los valores de los registros del conjunto de datos de entrenamiento. Cuando dichos valores son de tipo texto, se requiere transformar los mismos de forma que se les pueda aplicar una medida de distancia. En la siguiente sección se detallan los conceptos alrededor de la representación de documentos de texto.

2.3. Representación de documentos mediante Vectores de Palabras

Se tiene un conjunto de datos D formado por m documentos de texto en los cuales existe un total de n palabras diferentes. La idea principal es representar cada documento como un vector $d = p_1, p_2, ..., p_n$, donde el elemento p_i corresponde al

número de ocurrencias de la i-ésima palabra en dicho documento. [38]. De esta forma, un conjunto de documentos puede verse como la matriz en 2.2, siendo cada entrada p_{ij} el valor de la frecuencia del término j en el documento i.

$$D = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \tag{2.2}$$

La representación vectorial de un documento puede incluir, en lugar del valor de la frecuencia, un peso correspondiente a cada palabra. La técnica conocida como TF-IDF (por los términos en inglés *Term Frequency - Inverse Document Frequency*) permite calcular un peso para cada palabra, combinando el valor de la frecuencia de un término en un documento, con el número de documentos que contienen dicho término con respecto al total de documentos [38]. TF-IDF busca darle más relevancia a aquellos términos que permitan representar fácilmente documentos específicos y darle menos peso a los términos irrelevantes [38].

Otras técnicas que se utilizan comúnmente al momento de transformar documentos en vectores, son la eliminación de palabras no relevantes (más conocidas como *stop words*), y la derivación de palabras (*stemming*) mediante la cual se elimina la inflexión de los términos con el fin de agruparlos semánticamente [38]. De esta forma, se eliminan palabras tales como pronombres, artículos y conjunciones que no aportan mucha información y palabras como *relation*, *related* y *relate* se transforman en la misma palabra raíz *relat*.

Como ya se mencionó, una vez obtenida la representación vectorial de los documentos, se pueden aplicar funciones de similitud sobre ellos. Esto se detalla en la siguiente sección.

2.4. Medidas de similitud de texto

Este trabajo se enfoca en las medidas de similitud de texto estadísticas o basadas en términos, en los cuales una hilera es dividida en términos individuales, tales como un vector de palabras, y esta división es utilizada para realizar cálculos y comparaciones [43]. Un algoritmo de clasificación como KNN puede hacer uso de éstas métricas para calcular distancias entre los elementos.

Entre las métricas más comunes para comparación de vectores de palabras se encuentran la similitud *Jaccard* y la distancia de Coseno [37, 43]. Sin embargo, también existen otros enfoques basados en conceptos probabilísticos no tan comunes pero que también han sido utilizados para comparar documentos, tal como la divergencia Kullback-Leibler [5, 31]. Todos estos métodos se explican en las siguiente secciones.

2.4.1. Distancia de Coseno

Esta distancia mide el grado de similitud entre dos documentos d_1 y d_2 utilizando el coseno del ángulo formado por las respectivas representaciones vectoriales, tomando en cuenta la frecuencia de cada palabra en los documentos, tal como se muestra en la fórmula 2.3 [37].

$$cos(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\| \vec{d_1} \| \ \| \vec{d_2} \|} \tag{2.3}$$

En el apéndice B.1 se puede observar un ejemplo simple de cómo la función coseno es aplicada para similitud entre dos documentos de texto.

2.4.2. Similitud de Jaccard

Si se consideran dos vectores de palabras d_1 y d_2 como conjuntos de términos (no se toma en consideración la frecuencia), se define la similitud Jaccard en 2.4 como el tamaño de la intersección entre d_1 y d_2 dividido entre el tamaño de la unión de los mismos conjuntos [2].

$$jaccard(\vec{d_1}, \vec{d_2}) = \frac{| \vec{d_1} \cap \vec{d_2} |}{| \vec{d_1} \cup \vec{d_2} |} \tag{2.4}$$

2.4.3. Divergencia Kullback-Leibler

Considerando un documento como una distribución de probabilidad de palabras representado en un vector, se puede utilizar la divergencia Kullback-Leibler (KL), también conocida como la entropía relativa, para medir las diferencias entre dos distribuciones de probabilidad [18]. De forma que si n es el total de palabras en una colección de documentos, dos documentos d_1 y d_2 pueden representarse como distribuciones de probabilidad $\vec{d_i} = p_{i,1}, p_{i,2}, \ldots, p_{i,n}$ donde el valor $p_{i,j}$ indica la probabilidad de que la palabra j pertenezca al documento d_i para $i \in \{1, 2\}$ y $1 \leq j \leq n$, entonces la divergencia KL se calcula en la ecuación 2.5.

$$D_{KL}(\vec{d_1} \parallel \vec{d_2}) = \sum_{j=1}^{n} p_{1,j} \times log(\frac{p_{1,j}}{p_{2,j}}) \tag{2.5}$$

La divergencia KL ha mostrado ser efectiva en *clustering* de documentos y al no ser una función simétrica se debe combinar con otros métodos para lograr sintetizar los resultados de $D_{KL}(\vec{d_1} \parallel \vec{d_2})$ y $D_{KL}(\vec{d_2} \parallel \vec{d_1})$, como por ejemplo, la utilización de una función de promedio ponderado [18].

Dados los algoritmos, métricas y técnicas de representación de documentos descritos hasta el momento, se puede hacer uso de métodos agregados, tales como los que se describen en la siguiente sección.

2.5. Métodos agregados

La idea de los métodos agregados es complementar los resultados de diferentes clasificadores con el fin de mejorar la precisión, de forma que al ser combinadas las predicciones se tome una decisión global basada en ciertas estrategias [35].

En [15] se establece que una condición necesaria y suficiente para que un método agregado sea más exacto que un clasificador simple, es que los clasificadores base sean exactos y diversos (donde exactos quiere decir que tienen una tasa de error

mejor que adivinar al azar y donde diversos quiere decir que los métodos cometen errores distintos).

A continuación se describe la técnica de agregación utilizada en este proyecto.

2.5.1. Bagging

El método conocido como *Bagging* consiste en entrenar un algoritmo utilizando diferentes muestras aleatorias, elegidas del conjunto de datos de entrenamiento original con reemplazo [35]. Aplicado a clasificadores basados en distancias de similitud de texto, el flujo general se puede observar en el algoritmo 2, donde el método *ObtenerMuestraAleatoria* crea una muestra aleatoria de tamaño fijo tomada del conjunto E, $\vec{c}$ se encarga de ir almacenando los clasificadores formados a partir de aplicar el algoritmo de clasificación y la medida de similitud sobre cada muestra, y finalmente se retorna una función agregada $A(.)$ que elige el resultado obtenido por la mayoría de clasificadores [35].

Algoritmo 2: Método *Bagging* aplicado a algoritmo de clasificación con medida de similitud

Input: *E, T, S, C*
Sea E el conjunto original de datos de entrenamiento;
Sea T el tamaño de la agregación;
Sea S la medida de similitud de texto;
Sea C el algoritmo de clasificación;
for $t \leftarrow 1$ **to** T **do**
    $M_t \leftarrow$ ObtenerMuestraAleatoria(E);
    $c_t(.) \leftarrow \mathrm{C}(M_t, S)$;
Output: $A(.) \leftarrow$ ObtenerCategoriaPorVotacion($\vec{c}$)

El método *bagging* permite disminuir los errores de varianza en un modelo de clasificación (es decir, cuando pequeños cambios en el conjunto de datos de entrenamiento producen grandes cambios en el clasificador), pero no mejora errores de sesgo, que generalmente están relacionados a la calidad de los datos [2].

2.6. Modelado de temas

El modelado de temas ha surgido como una técnica muy robusta para estructurar colecciones de documentos, haciendo uso de modelos probabilísticos para encontrar patrones semánticos ocultos [38]. Además, representa un método alternativo que logra reducción dimensional y tiene muchas ventajas con respecto a otros modelos de análisis semántico [2].

En este trabajo se utilizó la representación de documentos en temas obtenida mediante la Asignación de Dirichlet Latente, la cual se explica a continuación.

2.6.1. Asignación de Dirichlet Latente

La Asignación de Dirichlet Latente (LDA por sus siglas en inglés) es un modelo probabilístico utilizado para clasificar documentos en temas (o categorías) tomando en cuenta los siguientes conceptos: un mismo documento puede tener varios temas latentes (puede pertenecer a más de una categoría) y cada tema puede ser representado como una distribución de palabras [5]. En los apéndices B.2 y B.3 se presentan más detalles sobre la distribución Dirichlet, la cual es considerada una distribución de probabilidad aplicada sobre funciones de probabilidad [13]. En este caso, una asignación de probabilidades de temas latentes en un documento, donde cada tema, a su vez, es una distribución de probabilidades de palabras.

La idea de LDA es lograr reducción de dimensionalidad (con respecto a las representaciones tradicionales de vector de palabras) y fácilmente poder asignar probabilidades a documentos nuevos que no fueron parte del conjunto de datos de entrenamiento [2].

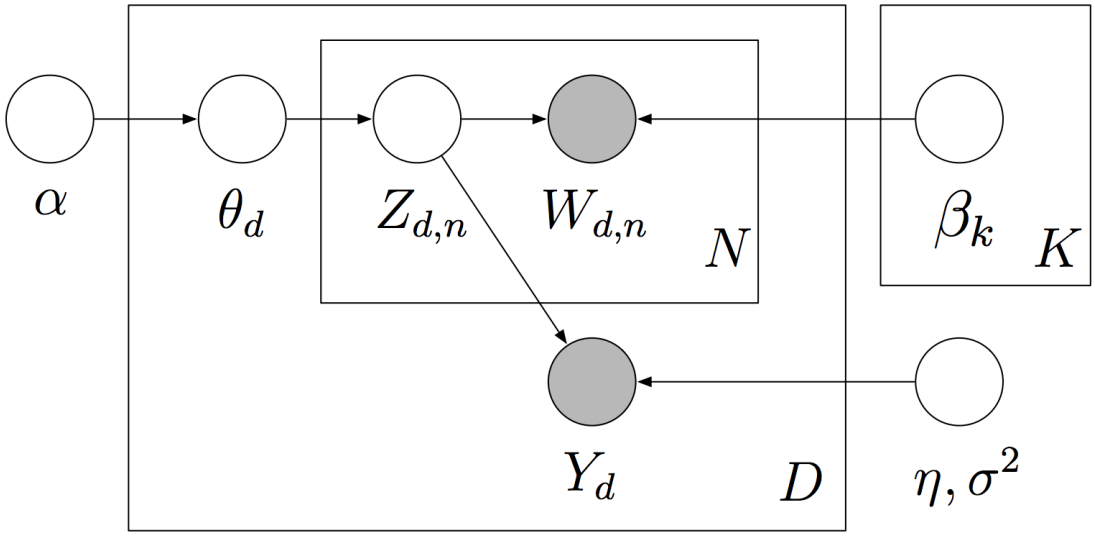
Aunque LDA fue desarrollado originalmente para el descubrimiento de temas latentes de manera no supervisada [7], este trabajo explora la variante supervisada de LDA que se propone en [30]. En LDA supervisado, si se tienen K temas, $\beta_{1:K}$

(donde cada β_k es un vector de probabilidades de palabras por tema), un parámetro de Dirichlet α, y parámetros de la variable de respuesta η y σ^2, se asume que cada documento de texto se genera de la siguiente manera [30]:

1. Se obtienen las proporciones de temas $\theta \mid \alpha \sim Dir(\alpha)$.
2. Para cada palabra:
   - Se asigna un tema $z_n \mid \theta \sim Mult(\theta)$.
   - Se obtiene una palabra $w_n \mid z_n, \beta_{1:k} \sim Mult(\beta_{z_n})$.
3. Se obtiene la variable de respuesta (clase) $y \mid z_{1:N}, \eta, \sigma^2 \sim N(\eta^T \bar{z}, \sigma^2)$.

Este modelo generativo se puede observar gráficamente en la figura 2-1. Cada nodo en la figura denota una variable aleatoria, mientras que las aristas representan dependencias. Las cajas indican repetición para D documentos, K temas y N palabras [38].

Figura 2-1: Representación gráfica del modelo LDA supervisado [30].



En LDA la representación de un documento d_i no se hace mediante un vector de palabras, sino mediante un vector de k temas latentes $\vec{d_i} = p_{i,1}, p_{i,2}, \ldots, p_{i,k}$ donde el valor $p_{i,j}$ indica la probabilidad de que d_i pertenezca al tema j para $1 \leq j \leq n$

[5, 2]. Posteriormente, la similitud entre las dos distribuciones de probabilidad puede calcularse haciendo uso de alguna medida de similitud. Esta investigación se enfoca únicamente en dicha representación vectorial de documentos y no en el algoritmo completo que involucra LDA ni en su modelo de predicción.

En el siguiente capítulo se describen los aspectos generales de esta investigación.

Capítulo 3

Descripción de la Investigación

3.1. Planteamiento del Problema

Actualmente, es común encontrar bases de datos en las cuales existen instancias o tuplas incompletas, es decir, que contienen valores faltantes en alguno de los atributos. Existen múltiples métodos para la imputación de dichos valores, por ejemplo, métodos que involucran algoritmos de aprendizaje máquina para predecir el valor faltante, basados en los valores de los atributos completos de la instancia [40, 19].

Este proyecto se enfoca en aquellos casos donde los atributos completos son de tipo texto libre y en los cuales la imputación se realiza mediante algún algoritmo de clasificación y aprendizaje supervisado. En este caso, el algoritmo utiliza un conjunto de datos de entrenamiento formado por instancias sin valores faltantes, donde la etiqueta de clase corresponde al valor del atributo que eventualmente se intentará imputar en registros sin dicho valor.

El hecho de que los atributos a considerar sean textuales representa varios retos para el algoritmo de clasificación. Métodos típicos de estimación o regresión lineal utilizados para imputación de datos en series de tiempo y atributos numéricos quedan descartados [29]. Además, un texto libre implica una cadena de caracteres con valores no estandarizados [22]. Por ejemplo, si se trata de valores introducidos por

usuarios existe la posibilidad de que todas las instancias contengan diferentes valores para el mismo atributo.

En compañías grandes, donde se manejan cientos de bases de datos que almacenan información de transacciones generadas diariamente, este escenario resulta común. Por ejemplo, en el 2015, en Intel® se encontró que alrededor del 20 % de los perfiles de usuarios creados en el sitio web de empleos de la compañía, no contaban con un valor para el país de procedencia. Esto representó un grave problema debido a que toda la información de perfiles de candidatos estaba siendo migrada a un nuevo sistema, en el cual el valor del país era requerido. Aunque una persona podría inferir dicho valor basándose en otros atributos como el nombre del candidato, la ciudad, el código postal o la dirección, no se contaba con suficientes recursos para cubrir los casi 200 mil registros con valores faltantes. El hecho de que los atributos completos fueran de tipo texto, libremente introducido por los candidatos al registrarse, representaba un reto computacional y algorítmico si se intentaba automatizar el proceso para imputar el valor del país. Por lo tanto, la decisión tomada consistió en descartar cualquier registro con valores faltantes, ocasionando pérdida de alrededor de un 20 % de los datos durante la migración [1].

En situaciones como esta, la minería de texto juega un papel importante y actualmente se han propuesto diferentes métodos para la clasificación que involucran técnicas de representación de documentos, medidas de similitud léxica y semántica para comparar valores de texto y algoritmos de aprendizaje máquina [43]. Documentos de texto similares pueden ser agrupados en categorías por el algoritmo de clasificación.

Existen muchos escenarios, tal como el caso de la base de datos de Intel®, en los que no hay una solución óptima y, por lo tanto, resultan en problemas interesantes de explorar. Por ejemplo, casos en los cuales se cuenta con textos muy cortos donde

[1] Información obtenida por el autor al desempeñarse como desarrollador de aplicaciones en el área de Recursos Humanos en Intel® Costa Rica

no existe un contexto ni suficiente información para inferir la semántica, o cuando un mismo texto involucra diferentes tipos de entidades como acciones, nombres propios o fechas que no pueden compararse de la misma manera [16, 31].

Es precisamente este tipo de problemas el que se estudia en este trabajo, combinando técnicas de representación de documentos y funciones de similitud de texto, buscando mejorar la exactitud al momento de imputar datos faltantes mediante clasificación.

En la siguiente sección se describe de forma general el trabajo de investigación desarrollado.

3.2. Descripción general de la investigación

Dado el problema planteado anteriormente, el presente proyecto buscó evaluar la exactitud obtenida por diferentes métodos al momento de clasificar documentos de texto. Esto con el fin de lograr un mayor nivel de acierto al imputar valores faltantes bajo las circunstancias descritas en el problema. Para ello, se realizaron varios experimentos que combinaban diferentes técnicas de representación de documentos y métricas de similitud de texto. Esta combinación se realizó mediante el algoritmo KNN de clasificación de aprendizaje supervisado. De igual manera, se incorporó el método agregado *bagging* asumiendo que el mismo lograría aumentar la robustez y el rendimiento de los algoritmos [35].

El trabajo principal consistió en la implementación de un programa en Python capaz de manejar el algoritmo KNN con el método agregado *bagging*. Este programa permitió combinar técnicas de representación de documentos y métricas de similitud de texto (vector de palabras, LDA, similitud coseno, distancia *Jaccard* y divergencia KL). Los algoritmos se aplicaron sobre conjuntos de datos compuestos por registros de dos atributos: uno de tipo texto y otro de tipo categórico. De esta forma, los algorit-

mos inferían la categoría basados en el atributo de texto. Los datos utilizados fueron tomados del repositorio Reuters, el cual está formado por conjuntos de documentos de los cuales se pueden extraer diferentes categorías y se describirán detalladamente más adelante.

Para cada uno de los experimentos se recolectaron datos referentes a las imputaciones realizadas en los conjuntos de datos de prueba, así como los valores de la exactitud, precisión, sensibilidad y puntaje F1.

3.3. Trabajos Relacionados

Actualmente se han propuesto diversos algoritmos en al campo de la minería de datos y del aprendizaje máquina para resolver problemas de valores faltantes mediante clasificación. En el caso del presente proyecto, se buscó implementar algoritmos que permitieran encontrar relaciones entre atributos de tipo texto y atributos de tipo categórico. En esta sección se resume el trabajo realizado por diferentes autores alrededor de estos temas.

En [40, 42, 19] se habla de varios métodos de clasificación supervisada tales como *hotdeck*, KNN y Árboles Aleatorios. Al ser supervisadas, dichas técnicas requieren de un conjunto de datos entrenamiento etiquetado y además se caracterizan por basarse en medidas de similitud que permiten definir las relaciones entre los elementos a clasificar. Existen también algoritmos basados en modelos estadísticos tales como la imputación *likelihood* (que utiliza la técnica de $Expectation - Maximization$) [3] y los modelos bayesianos (*Naive Bayes* y *Bayesian Linear Regression*) [29]. Otras técnicas populares incluyen modelos matemáticos que intentan minimizar errores o ajustar una función basados en conocimiento obtenido a partir de datos de entrenamiento, como por ejemplo, Redes Neuronales Artificiales [33] y Máquinas de Soporte Vectorial [17]. De todos estos algoritmos, los primeros son los que más se ajustan al

problema planteado en este proyecto, dado que dependen directamente de medidas de similitud.

En términos generales, cuando se habla de análisis de documentos de texto a gran escala, la literatura se enfoca en dos tipos de métodos: los lingüísticos y los estadísticos [38].

En los métodos lingüísticos, la idea es manejar el texto basado en procesamiento de lenguaje natural, tomando en cuenta la representación semántica y las relaciones que se expresan entre las palabras en un modelo de lenguaje [38]. De esta manera, diversos autores han incorporado el factor semántico al momento de comparar documentos, siendo necesaria la integración de una fuente externa de conocimiento como Wikipedia o técnicas ontológicas [36]. En [25, 11, 28] se hace uso de $WordNet$, una de las bases de datos léxicas del idioma inglés más utilizadas para determinar similitud semántica entre documentos. En estos estudios, la similitud entre palabras está determinada por el grado de traslape en los significados (dado que una palabra puede tener múltiple semántica) y la distancia entre los dos términos al ser representados en un grafo de hiperónimos (Por ejemplo, la palabra "animal" y la palabra "perro" están relacionadas debido a que en el grafo taxonómico una contiene a la otra) [11]. Además de $WordNet$, otros autores han utilizado el repositorio *Google tri-grams* como tesauro del idioma inglés para similitud semántica [10, 37].

A pesar de que estos métodos lingüísticos pueden llegar a formar representaciones más expresivas del texto, también agregan más complejidad al proceso. Esto debido a la dificultad que supone la construcción de un modelo semántico y a la dependencia con el contexto específico de cada conjunto de datos [38]. Este trabajo se enfocó, principalmente, en métodos estadísticos que involucran la representación matemática del texto sin tomar en cuenta la semántica ni las propiedades del lenguaje [38]. El proceso general consiste en elegir una técnica para la representación de dos documentos de texto y luego aplicar una función o métrica sobre ambas representaciones para

calcular la similitud.

El vector de palabras es la técnica de representación más común, utilizada en la mayoría de la literatura consultada [2, 37, 24, 44, 38, 25]. Ésta consiste en representar cada documento de texto como un vector, donde cada entrada corresponde a un término y el valor de la entrada es la frecuencia de ese término en el documento [2]. Otra técnica común de representación es la denominada $TF - IDF$ utilizada en [4, 37], en la cual se le asignan pesos a los términos de un vector, de modo que se le resta relevancia a palabras que aparecen con mucha frecuencia en el conjunto de datos, y se le da importancia a aquellas con pocas ocurrencias.

En [25, 5] se hace uso de LDA, un modelo probabilístico generativo que se basa en dos conceptos principales: 1) un mismo documento de texto puede tener muchos temas latentes y 2) cada tema está formado por una distribución de palabras. De esta forma, en lugar de un vector de palabras, un documento puede ser representado por un vector de temas distintos con las probabilidades de pertenecer a cada uno. Con respecto al vector de palabras, la representación LDA logra reducción dimensional (dado que el tamaño no depende del total de términos distintos en todo el conjunto de datos), considera co-ocurrencia de palabras y sinónimos entre documentos (incorporando así el elemento semántico) y toma en cuenta todo el conjunto de datos al utilizar todas las categorías [5].

La variante supervisada de LDA se propone en [30], de forma que se incorpora una variable de respuesta (clase) al momento de calcular el modelo de asignación de temas y palabras. En dicho trabajo, LDA es utilizado para inferir calificaciones de películas y páginas web basándose en el texto de los comentarios de los usuarios. LDA supervisado también ha sido utilizado para clasificación y etiquetado de imágenes, donde cada imagen es representada como un documento de texto formado por palabras claves [9].

Con respecto a las métricas de similitud de texto, la literatura es bastante extensa, siendo la similitud de Coseno una de las técnicas más utilizadas, tal como en [26, 37]. En [5], los autores combinan la representación LDA con la función Coseno para calcular la similitud entre publicaciones basadas en el *abstract*, el título y los autores. Otras medidas utilizadas tanto para clasificación como para agrupamiento de documentos de texto, incluyen la similitud de *Jaccard* [26, 37] y otras que se basan en comparación de hileras de texto como la distancia *levenshtein* [14, 41] y la medida *longest common subsequence* [20, 37].

Huang [18] hace uso de la divergencia KL para determinar la distancia entre dos documentos, suponiendo que la representación está basada en distribuciones de probabilidad. Esta métrica también es conocida como entropía relativa y, al no ser simétrica, se realiza un cálculo de promedio ponderado para obtener el valor final. Estudios realizados en [18, 31], muestran que la divergencia KL supera considerablemente a la distancia coseno, la similitud *Jaccard* y la distancia euclidiana al momento de calcular el grado de pureza y entropía sobre ciertos conjuntos de datos de tipo texto.

En las siguientes secciones se detallan aspectos claves del proyecto tales como la hipótesis de investigación, los objetivos y la descripción de los experimentos realizados.

3.4. Hipótesis

Considerando la definición del problema planteado y el marco teórico descrito anteriormente, se plantea la siguiente hipótesis:

En la imputación de valores faltantes basada en clasificación de atributos de texto, un algoritmo agregado que utilice técnicas de representación de documentos y medidas de similitud de texto: LDA y divergencia KL, obtiene una exactitud mayor que el mismo algoritmo agregado cuando utiliza métodos tradicionales (vector de palabras, similitud *Jaccard* y distancia

Coseno).

3.5. Objetivos

A continuación se define el objetivo general y los objetivos específicos.

3.5.1. Objetivo general

El objetivo de la presente investigación es el siguiente:

- Evaluar el uso de distintas métricas de distancia de texto en un algoritmo agregado para la imputación de valores faltantes mediante clasificación.

3.5.2. Objetivos específicos

Los objetivos específicos se listan a continuación:

1. Implementar un algoritmo para la imputación de valores faltantes utilizando las técnicas LDA y divergencia KL de forma agregada para clasificar valores de tipo texto.

2. Evaluar el algoritmo implementado comparando resultados obtenidos en experimentos que utilicen dicho algoritmo y algoritmos agregados que utilicen medidas de similitud de texto tradicionales.

3. Explicar los resultados obtenidos luego de la ejecución de los experimentos de manera que se logren destacar las diferencias entre la exactitud obtenida al utilizar los diferentes algoritmos.

3.6. Resumen de los experimentos

En la presente sección se resume el diseño de experimentos, para el cual se siguieron los lineamientos planteados en [32], que incluyen la declaración del problema, la definición de los factores y niveles, las variables de respuesta y la prueba de hipótesis.

3.6.1. Declaración del Problema

Determinar si la utilización de distintas métricas de similitud de texto combinadas mediante algún método agregado mejoran el nivel de acierto en procesos de imputación de valores faltantes.

3.6.2. Factores y Niveles

Los siguientes factores y sus respectivos niveles fueron considerados durante la ejecución de los experimentos:

1. Representación de los documentos de texto
    a. Vector de Palabras
    b. Asignación de Dirichlet Latente
2. Medida de similitud de texto
    a. Divergencia Kullback-Leibler
    b. Similitud Jaccard
    c. Distancia Coseno
4. Conjunto de datos
    a. *Reuters*: Imputación del tema basado en el contenido textual de la noticia
    b. *Reuters*: Imputación del lugar al que hace referencia una noticia basado en el texto

3.6.3. Variables de Respuesta

En esta sección se detallan las variables a considerar para evaluar los experimentos de este proyecto, descritas en [43]. Éstas corresponden a la exactitud, precisión, sensibilidad y F1. De esta forma, si se tiene un conjunto de datos D, donde X es un subconjunto de D que contiene aquellos registros en los cuales el algoritmo imputó

los datos correctamente, I_c contiene las instancias en las cuales el algoritmo imputó el valor c y R_c es el conjunto de registros en los cuales el valor faltante realmente corresponde al valor c, entonces:

1. La **exactitud** determina el porcentaje de aciertos en un experimento tomando en cuenta el total de la población, usando la siguiente fórmula:

$$exactitud = \frac{\mid X \mid}{\mid D \mid} \tag{3.1}$$

2. La **precisión** en un determinado experimento y para una determinada etiqueta o clase c, representa la proporción de instancias de dicha clase correctamente clasificadas con respecto al total de instancias clasificadas bajo esa etiqueta por el algoritmo. La fórmula utilizada es la siguiente:

$$precision = \frac{\mid I_c \cap R_c \mid}{\mid I_c \mid} \tag{3.2}$$

3. La **sensibilidad (*recall*)** en un determinado experimento y para una determinada etiqueta o clase c, corresponde a la proporción de instancias de dicha clase correctamente clasificados sobre el total de instancias donde el valor correcto es esa clase:

$$sensibilidad = \frac{\mid I_c \cap R_c \mid}{\mid R_c \mid} \tag{3.3}$$

4. Y el **puntaje F1 macro ponderado (*F1 score*)** permite realizar un promedio ponderado de la precisión y de la sensibilidad mediante la siguiente fórmula:

$$F1 = \frac{2 * precision * sensibilidad}{precision + sensibilidad} \tag{3.4}$$

Cabe destacar que el análisis principal de este proyecto se basó en la variable de exactitud, sin embargo, se utilizó el valor de F1 (que implicó el cálculo de la precisión y de la sensibilidad) para complementar el análisis de los resultados. Dado que los experimentos realizaban imputación sobre múltiples clases, se utilizó la variante F1

macro ponderada, la cual promedia los valores F1 obtenidos para cada una de las clases.

3.6.4. Análisis estadístico

Para validar si las distribuciones obtenidas como resultado de los experimentos mostraron o no diferencias significativas de acuerdo a la estrategia utilizada, se emplearon bibliotecas de R para análisis estadístico.

En dicho análisis, se utilizó la prueba de normalidad Lilliefors (que está basada en la prueba Kolmogorov–Smirnov [27]) y se determinó que, en la mayoría de los casos a evaluar, los valores de exactitud y F1 estaban distribuidos normalmente. De esta forma, se procedió a utilizar ANOVA como método paramétrico para el análisis de las varianzas. El objetivo de ANOVA es analizar la relación entre una variable cuantitativa X y una variable cualitativa Y la cual puede tener k diferentes valores. Cada uno de estos valores cualitativos define una población cuyo valor es dado por la variable cuantitativa [6]. En otras palabras, se agrupan los valores de X en k poblaciones $x_1, \ldots, x_k$ y se desea determinar si X no varía con respecto a los valores de Y:

- **Hipótesis nula:** Las muestras pertenecen a poblaciones idénticas (X no varía según los valores de Y)
- **Hipótesis alternativa:** Al menos una muestra pertenece a una población diferente que las demás (X varía dado algún valor de Y).

Adicionalmente, para los casos en los cuales se encontraron diferencias significativas en las medias, se utilizó TukeyHSD como prueba *post-hoc* para comparación múltiple de pares, de forma que se determinara específicamente cuál muestra o muestras pertenecían a otra población [1].

Para aquellas muestras que no pasaron la prueba de normalidad, se utilizó el método no-paramétrico conocido como la prueba Kruskal-Wallis, el cual no necesita

supuestos de que la población sigue una distribución normal [23]. En estos casos, se utilizó la prueba de Nemenyi para comparación múltiple de pares [34].

3.7. Detalles de los experimentos

A lo largo de esta sección se profundiza en los detalles técnicos y de proceso relacionados a la preparación, implementación y ejecución de los experimentos realizados.

3.7.1. Ambiente de Desarrollo

Las herramientas utilizadas para preprocesamiento, implementación, ejecución y validación de los experimentos debían ser capaces de ajustarse a los objetivos planteados, de forma que se facilitara el desarrollo y la configuración de los diferentes métodos a utilizar. A continuación se detallan los aspectos técnicos que fueron indispensables para lograrlo.

Sistema Operativo y hardware

Se trabajó con un servidor virtual con 2 procesadores de 2GHz cada uno y 4 GB de memoria RAM. La idea fue tener una máquina encendida en todo momento y con suficiente poder de procesamiento para mantener corriendo algoritmos de preprocesamiento y clasificación de texto sin interrupción y con poca intervención humana. Dicho servidor cuenta con sistema operativo Linux (distribución Ubuntu 16.04). Para algunos casos donde se requería manipulación de archivos mediante programas simples y rápidos de ejecutar, se utilizó una computadora personal con sistema operativo Windows 10.

Lenguajes de programación

- Los conjuntos de datos originales se obtuvieron en formato de archivos XML, por lo que el procesamiento inicial se realizó mediante un programa en lenguaje C# usando Visual Studio Community 2015. Con este programa se logró transformar

los datos a formato CSV. También se utilizó C# para preparar los archivos en el formato requerido por el algoritmo de LDA supervisado.

- La representación de documentos en vectores de palabras se obtuvo mediante un programa desarrollado en R versión 3.2.3. El análisis estadístico también se realizó con paquetes de R.

- La implementación de los algoritmos principales (KNN, *bagging* y métricas de distancia de texto) se realizó en el lenguaje Python versión 3.5.2. Este mismo lenguaje se utilizó para el cálculo de las variables de respuesta luego de obtener los resultados.

- Para LDA supervisado se utilizó un programa implementado en C++ basado en los trabajos de [30, 9] [2].

- Otros lenguajes de programación complementarios, utilizados para transformación y persistencia de los datos a lo largo del proceso, fueron PHP 7.0.15 y MySQL 5.7. También se utilizó el paquete D3 de Javascript para generación de gráficos de resultados.

Bibliotecas de R

- **RTextTools versión 1.4.2**. Contiene las operaciones necesarias para transformar un documento de texto en su representación vectorial [21].

- **Caret versión 6.0-73**. Utilizado para particionar los datos en conjuntos de prueba y de entrenamiento de forma balanceada.

- **PMCMR versión 4.1**. Requerido para la ejecución de pruebas Kruskal-Wallis y comparación múltiples de pares.

- **Nortest versión 1.0-4**. Prueba de normalidad Lilliefors (Kolmogorov-Smirnov).

[2] http://www.cs.cmu.edu/ chongw/slda/

Bibliotecas de Python

- ***scikit-learn* versión 0.18**. Esta biblioteca contiene implementaciones de KNN, incluida $KNeighborsClassifier$, la cual permitió configurar el mismo algoritmo para diferentes métricas de comparación de tipo $DistanceMetric$. Esta clase dispone de varias medidas de similitud ya implementadas, y a su vez permitió la creación de funciones personalizadas.

- ***SciPy* versión 0.19.0**. Utilizada principalmente para el cálculo de la divergencia KL entre distribuciones de probabilidad.

3.7.2. Conjuntos de datos

Los datos utilizados en los experimentos pertenecen a la colección de Reuters para clasificación de texto y fueron obtenidos del repositorio público UCI[3] en formato de archivos XML (En el apéndice A.1 se muestra un ejemplo de registro tomado de estos archivos).

Para facilitar la ejecución de los experimentos, se crearon dos conjuntos de datos luego de transformar y compilar un subconjunto de los archivos XML tomados de la colección de Reuters. La transformación de los datos implicó lo siguiente:

1. Elección de un atributo categórico (los registros de Reuters contienen varios atributos categóricos como temas, lugares y personas).

2. Verificación de que el registro tuviera un único valor para la categoría elegida. Si el atributo categórico poseía múltiples valores entonces se descartaba. Lo mismo si no había un valor del todo.

3. Combinación de los atributos de texto en uno solo (En este caso, se combinó el título y el cuerpo del artículo).

[3]http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection

Esta transformación dio como resultado dos conjuntos de datos en archivos con formato CSV, que para efectos de esta investigación se nombraron reuters-1 y reuters-2:

- **reuters-1**. Contiene dos atributos, uno de tipo texto y otro de tipo categórico. Las categorias corresponden a 45 valores del atributo *topics* tomados del conjunto de datos Reuters original. Los valores de las clases y la distribución de registros por categoría se muestran en el apéndice A.2.

- **reuters-2**. De igual manera, contiene un atributo de tipo texto y otro de tipo categórico. En este caso las categorías corresponden a 60 valores diferentes del atributo *places* tomados de la colección Reuters original. Más detalles sobre las clases y la distribución de registros por categoría se muestran en el apéndice A.3.

Estos conjuntos de datos fueron elegidos para la investigación debido a que contienen registros con atributos de tipo texto ya categorizados (facilitando el proceso al no tener que invertir tiempo en etiquetado de los datos), y cumplen con las características necesarias para probar los algoritmos implementados. Aunque la distribución de clases es completamente desbalanceada (muy pocas clases contienen la mayoría de los registros), se decidió mantener los datos de esa forma ya que este mismo comportamiento se observó en los problemas reales que motivaron el desarrollo de este proyecto.

3.7.3. Implementación de los métodos de representación de datos

Una vez obtenidos los archivos en formato CSV con los conjuntos de datos, se requirió transformar cada documento en una representación vectorial. Las tres representaciones de datos consideradas en esta investigación fueron los vectores de palabras simples, vectores de palabras utilizando TF-IDF y vectores de temas usando LDA.

Los vectores de palabras se implementaron utilizando el paquete RTextTools de R, el cual provee funciones que permiten transformar fácilmente un conjunto de documentos de texto en una matriz. La misma herramienta se configuró para remover *stop words* y números, y realizar *stemming* de palabras. El proceso se ejecutó dos veces para cada conjunto de datos, uno almacenando las frecuencias de las palabras y otro calculando los pesos TF-IDF.

Como resultado, cada conjunto de documentos fue transformado en dos matrices, donde cada fila corresponde a un documento de texto y existe una columna por cada palabra del conjunto de datos. Para reuters-1 se generaron matrices de 1063 documentos x 6444 palabras, mientras que para reuters-2 las matrices fueron de 1641 documentos x 8666 palabras.

En este punto del proceso, también se realizó el particionamiento de los datos en un conjunto de 25 % de datos de prueba y un conjunto de 75 % de datos de entrenamiento, utilizando la biblioteca Caret de R. El hecho de realzar esta tarea de forma separada y previo a la ejecución de los algoritmos de clasificación, permitió asegurarse de que las diferentes combinaciones utilizadas en los experimentos trabajaran bajo las mismas condiciones (mismos datos de entrenamiento y datos de prueba para todos los experimentos). Se crearon también particiones de 50 %-50 % y 80 %-20 %, sin embargo, para efectos de análisis de resultados se trabajó únicamente con las de 75 %-25 %.

Para el caso de las representaciones mediante LDA, se utilizó un algoritmo de LDA supervisado implementado en C++. Este programa requería un archivo de entrada con un formato específico de palabras y frecuencias por documento, por lo que se aprovecharon las representaciones vectoriales creadas en R para generar dicho archivo. El programa funciona mediante dos fases:

- En la primera fase se realiza la estimación de los valores del modelo. De esta forma, el programa ejecuta una serie de iteraciones con el fin de crear un modelo

basado en los parámetros definidos y en un conjunto de datos de entrenamiento (textos con sus respectivas etiquetas). Este modelo es dado en forma de varios archivos de texto y, para efectos de esta investigación, únicamente interesaba el archivo de asignación de temas. Este se puede ver como una matriz de probabilidades, donde cada fila corresponde a un documento del conjunto de datos de entrenamiento y cada columna corresponde a un tema.

- Para la segunda fase, el programa requiere del modelo generado en la fase 1, además de un conjunto de datos de prueba en el mismo formato y sin etiquetas. El objetivo principal de esta etapa es generar la clasificación de los documentos de prueba, por lo que se realizó una modificación al código para que también generara un archivo con la asignación de temas para los datos de prueba. La idea fue evitar sesgos al mantener separadas la asignación de temas para documentos de prueba (sin etiquetar) y para documentos de entrenamiento (etiquetados).

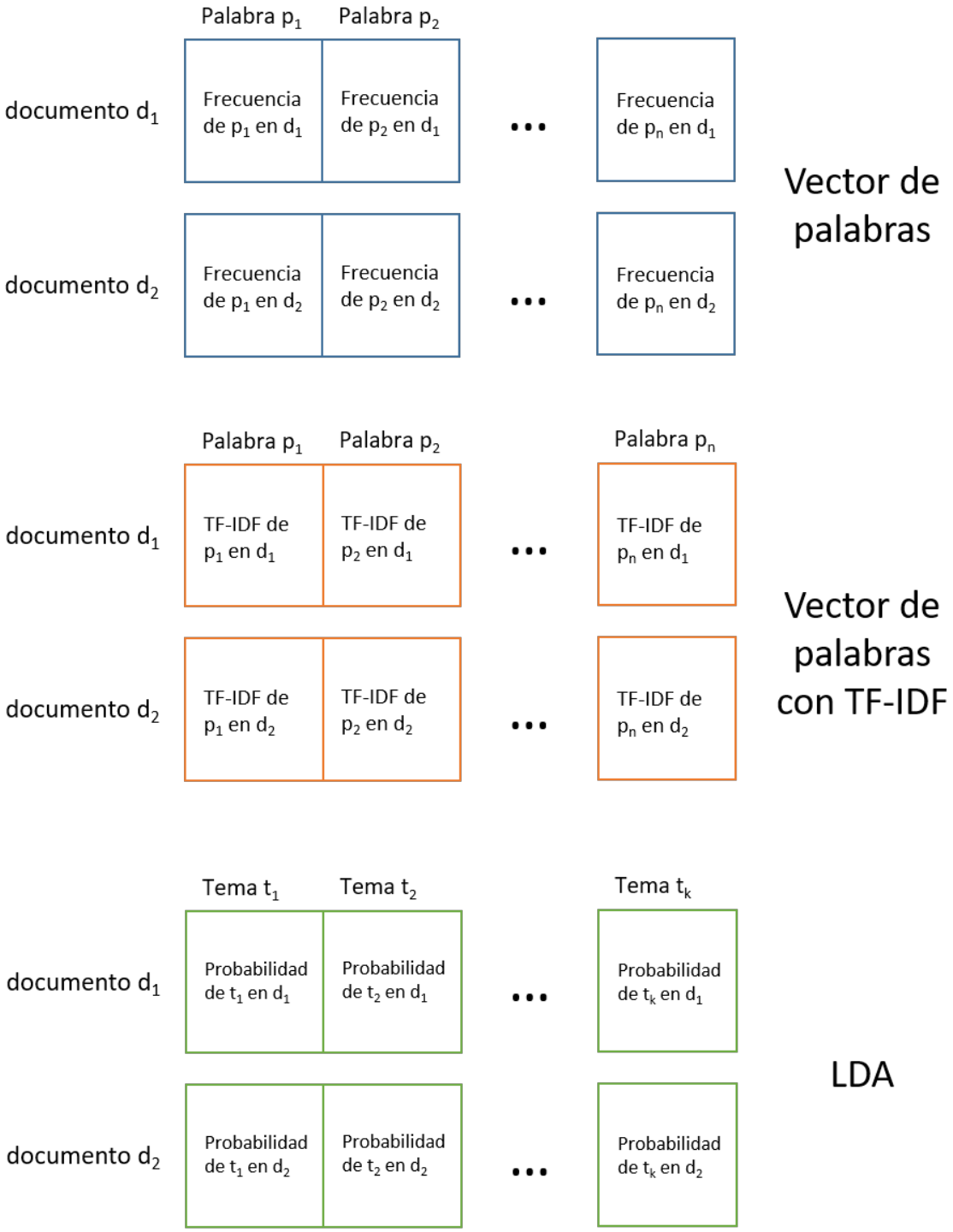
La representación del conjunto reuters-1 mediante LDA consistió en una matriz de probabilidades de 1063 documentos x 44 temas, mientras que para reuters-2 las dimensiones de la matriz fueron 1641 documentos x 59 temas.

La figura 3-1 muestra la idea general de las representaciones de documentos de texto implementadas, suponiendo que se tienen dos documentos d_1 y d_2, un total de n palabras y un modelo de LDA con k temas. La definición de parámetros para el programa de LDA supervisado se explica en la sección 3.7.5.

3.7.4. Implementación del algoritmo de clasificación y las métricas de distancia

El algoritmo fundamental para la ejecución de los experimentos de este proyecto fue KNN. Para ello se utilizó el módulo KNeighborsClassifier del paquete sklearn de Python. Esta implementación de KNN representó la mejor opción para la presente investigación, ya que cuenta con dos características clave:

Figura 3-1: Representación de dos documentos mediante vector de palabras simple, vector de palabras con TF-IDF y vector de temas con LDA.



1. Permite configurar la métrica de distancia de texto a utilizar para buscar los vecinos más cercanos con respecto a un documento.

2. Puede incorporarse fácilmente con métodos agregados del paquete sklearn.

Las métricas compatibles con este módulo deben ser de tipo DistanceMetric, sin embargo, se permite la definición de funciones personalizadas siempre y cuando reciban dos vectores numéricos como parámetro y se devuelva un valor numérico real. Basándose en esto, se implementaron las medidas de distancia de texto de la siguiente manera:

- Se mantuvo la distancia Jaccard incluida dentro del módulo DistanceMetric. Esta función requiere de dos vectores numéricos X y Y, ambos de n dimensiones y que son tratados como conjuntos. De forma que, para cualquier i donde $1 \leq i \leq n$, si $X[i] \neq 0$, significa que el elemento i pertenece al conjunto representado por X. De igual manera, si para algún valor de i se tiene que $Y[i] \neq 0$, significa que i pertenece al conjunto representado por el vector Y. Al identificar los elementos de cada conjunto, la función puede aplicar la fórmula 2.4. La distancia Jaccard se aplicó únicamente sobre conjuntos de datos representados mediante vectores de palabras usando frecuencias. Por definición, el valor de retorno está normalizado entre 0 y 1, por lo que no se requirió de alguna transformación adicional.

- Para la distancia Coseno se utilizó el módulo *cosine_similarity* que forma parte de las métricas del paquete sklearn. De igual manera, esta función requiere de dos vectores numéricos X y Y sobre los cuales se aplica la ecuación 2.3. Dado que esta métrica trabaja sobre el coseno del ángulo de los vectores y no realiza operaciones de conjuntos, el programa permite aplicar la distancia Coseno sobre cualquiera de las representaciones de documentos implementadas, incluyendo el vector de probabilidades de temas con LDA. Una vez calculado el valor de la similitud, éste se tuvo que normalizar a un valor entre 0 y 1 (la función coseno retorna valores entre -1 y 1) para luego ser transformada en medida de distancia. Este cálculo se muestra en la ecuación 3.5.

$$coseno\_dist(X, Y) = 1 - \frac{cos(X, Y) + 1}{2} \tag{3.5}$$

- Y por último, la Divergencia KL fue implementada como una función personalizada utilizando la métrica *entropy* del paquete *scipy*. Dicha métrica se encarga de normalizar los datos (en caso de no estarlo) entre 0 y 1, y el valor de entropía retornado es mayor para aquellos documentos que son más diferentes entre sí, incluyendo valores infinitos cuando se divide entre cero. Por esta razón, todas las probabilidades iguales a cero son sustituidas por valores muy cercanos a cero antes de aplicar la función y el valor retornado no requiere transformación al comportarse como una distancia.

Incorporación del método agregado *bagging*

Como se mencionó anteriormente, el módulo KNeighborsClassifier facilitó la integración de KNN con métodos agregados. En el caso de esta investigación, se implementó *bagging* utilizando el módulo BaggingClassifier incorporado dentro de los algoritmos agregados de sklearn. Este método funciona tal como se describe en el algoritmo 2 y requirió de los siguientes parámetros para su implementación:

- Un clasificador, el cual consistió en la instancia de KNN configurada con la respectiva métrica de distancia de texto y el valor de k, dependiendo del experimento.
- El tamaño de la muestra a utilizar para cada agregación.
- El número de atributos a considerar para cada agregación.

La implementación en Python de la lógica y los algoritmos discutidos en esta sección pueden apreciarse de manera más detallada en el apéndice F. En la siguiente sección se detallan los criterios utilizados para la elección de parámetros en cada uno de los algoritmos y experimentos que así lo requerían.

3.7.5. Seleccion de parámetros

En esta sección se describen los principales parámetros utilizados por los diferentes experimentos y se resumen los criterios para su escogencia. En la tabla del apéndice C.1 se muestran los diferentes valores seleccionados.

Para KNN, el parámetro fundamental fue el valor de k, el cual determina la cantidad de vecinos a considerar al momento de etiquetar un documento. Para esto, previo a la ejecución de todos los experimentos, se ejecutó un subconjunto de los mismos utilizando múltiples valores de k. Luego se calculó la exactitud obtenida para cada experimento y se observó su comportamiento. En el apéndice C.2 se muestran los gráficos obtenidos para los experimentos que involucraron la distancia coseno para reuters-1 y reuters-2 respectivamente. Como resultado, se observó una clara tendencia en la exactitud a disminuir conforme más grande es el valor de k. Por esta razón, se consideraron únicamente valores pequeños para k al ejecutar la totalidad de experimentos ($k = 1$, $k = 11$, $k = 21$ y $k = 31$). Una vez ejecutados los experimentos, se realizó un análisis estadístico sobre la exactitud obtenida usando k como variable cualitativa, cuyos resultados se muestran en el apéndice E.2. Esto último dio como resultado la selección de experimentos que utilizaron $k = 1$ y $k = 11$ para el análisis de resultados final.

En el caso de *bagging*, se requería definir el tamaño máximo de las muestras individuales en las cuales el algoritmo divide el conjunto de datos, además del número de atributos a considerar. Para este último, dado que ambos atributos: texto y clase eran relevantes, el valor se fijó en 1.0, representando la totalidad de atributos. El tamaño de las muestras se determinó de forma similar al valor de k, ejecutando un subconjunto de experimentos con los cuales se observó que la precisión no variaba mucho conforme se aumentaba dicho valor (la diferencia fluctuaba entre 0 y 0.1 puntos de exactitud), además de que al utilizar valores cercanos o superiores a 0.20 el programa fallaba de forma intermitente debido a errores de memoria insuficiente. De esta

forma, se seleccionó un tamaño de muestra relativamente pequeño: 0.15 (El conjunto de datos se divide aproximadamente en 7 muestras para cada predicción).

La selección de parámetros requeridos por el algoritmo LDA se realizó de la siguiente manera:

- El número de iteraciones y el error de convergencia en los procesos de *Expectation-Maximization*, se fijó con los valores por defecto de la biblioteca. Esto para facilitar la selección y por razones de desempeño (Se observó que al aumentar ligeramente el número de iteraciones de cada etapa, el tiempo de ejecución aumentaba considerablemente).

- El valor de alfa es fundamental para que el algoritmo determine qué tan similares o diferentes son los documentos entre sí, tal como se muestra en el gráfico del apéndice B.3. Dado que los conjuntos de datos contienen documentos muy variados, se decidió utilizar tres valores distintos de alfa para eventualmente comparar el comportamiento y los resultados obtenidos utilizando cada valor. Los valores utilizados fueron: 0.2 (valor pequeño que asume documentos muy diferentes entre sí y con pocos temas cada uno), 0.8 (valor cercano a 1, que asume que cada documento contiene una combinación de mucho temas) y un valor intermedio de 0.5.

- El número de temas (que corresponde al tamaño de los vectores generados por el modelo) se estableció de manera diferente dependiendo del conjunto de datos, asumiendo que el número de temas debía ser igual o similar al número de clases diferentes. Para reuters-1 el valor fue de 44 temas y para reuters-2 dicho valor fue 59, ambos correspondientes al número de etiquetas diferentes para los respectivos conjuntos de entrenamiento.

3.7.6. Obtención y análisis de resultados

Para el cálculo de resultados, se utilizó el módulo de métricas de sklearn [4], con el cual se obtuvo el valor de la exactitud, la precisión, la sensibilidad y el puntaje F1 de cada uno de los experimentos. Para facilitar el análisis de resultados, solamente se tomó en cuenta la exactitud y el valor de F1 (dado que este último representa el promedio ponderado entre la precisión y la sensibilidad). El hecho de que la imputación de datos en los experimentos se realizó para múltiples clases, el cálculo del puntaje F1 debía realizarse para cada una de las etiquetas diferentes en los conjuntos de datos, por esta razón se utilizó la modalidad F1 macro ponderada, la cual calcula un único valor para cada experimento realizando un promedio ponderado de los valores obtenidos para cada clase.

Toda la información obtenida como resultado de los experimentos fue almacenada en una base de datos MySQL para facilitar la interacción con la misma. Esta base de datos incluye, para cada experimento, los conjuntos de datos de entrenamiento y de prueba involucrados, las imputaciones realizadas por el algoritmo y los valores de la exactitud y el puntaje F1.

Con respecto al análisis de resultados, se realizaron pruebas estadísticas para validar si las distribuciones obtenidas como resultado de los diferentes experimentos, eran estadísticamente diferentes de acuerdo a la estrategia de representación de datos y métrica de distancia de texto empleada. Para ello, se utilizó una prueba de ANOVA tanto para la exactitud como para el puntaje F1, ambas con respecto a la combinación métrica-representación de datos. De manera complementaria, se emplearon otros métodos estadísticos descritos en la sección 3.6.4.

[4] http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

3.8. Alcance del proyecto

Los siguientes aspectos resumen el alcance del presente proyecto a ser considerado para la evaluación del mismo:

- Los programas implementados incluyen el algoritmo de clasificación KNN, las técnicas de representación de documentos Vector de Palabras y LDA, y las métricas de similitud de texto Coseno, *Jaccard* y divergencia KL. Cualquier otro método queda fuera del alcance de este proyecto.
- El método agregado implementado corresponde a *bagging* únicamente.
- Todos las técnicas, funciones y algoritmos implementados se diseñaron únicamente para documentos de texto escritos en idioma **inglés**.
- La implementación de cualquier técnica que involucre representación de documentos, algoritmos complejos o modelos matemáticos y probabilísticos se realizó por medio de bibliotecas existentes. No se desarrolló ningún algoritmo desde cero, aunque sí se requirió hacer algunos ajustes personalizados en varios paquetes.
- El análisis de resultados se basó únicamente en las métricas de exactitud y puntaje F1.
- Los entregables a aportar son: programas implementados con los métodos propuestos mediante bibliotecas, conjuntos de datos preprocesados, documento de resultados obtenidos luego de la ejecución de los experimentos, documento con el análisis estadístico y la explicación de los resultados obtenidos con el fin de destacar los principales hallazgos encontrados durante la investigación.
- Si bien la lista de entregables no incluye un artículo científico, existe el acuerdo moral con el tutor de preparar un artículo científico en inglés para ser presentado a una revista que trate un tema afín, esto como máximo en los siguientes tres meses a la defensa.

- Métodos que involucren similitud de texto basada en semántica, técnicas lingüísticas, bases de conocimiento o secuencias de caracteres quedan fuera del alcance de este trabajo.

- El preprocesamiento de los datos se realizó mediante bibliotecas existentes para dicho propósito. Las técnicas se limitaron a la eliminación de palabras no relevantes (*stop words*), la derivación de palabras (*word stemming*) y la ponderación TF-IDF cuando fue necesario.

- La implementación de los algoritmos no se realizó con fines comerciales ni patentes que impidan el uso de los aportes generados en el presente proyecto.

Capítulo 4

Resultados

En este capítulo se presentan los resultados y su respectivo análisis. Para ello, se utilizan los datos mostrados en la tabla 4.1, que contiene las métricas obtenidas en cada uno de los experimentos para los dos conjuntos de datos, los valores de $k = 1$ y $k = 11$, así como las diferentes combinaciones de representación de datos y métrica de distancia de texto.

Como se ha mencionado anteriormente, cada experimento involucró un método de representación de datos, una medida de distancia de texto y el algoritmo KNN para clasificación. Sobre la tabla 4.1 cabe destacar lo siguiente:

- La representación de datos mediante vectores de palabras se separó en dos grupos: con TF-IDF y sin TF-IDF. Esto debido a que la distancia Coseno puede obtener diferentes resultados para cada grupo, mientras que Jaccard no toma en cuenta frecuencias ni pesos, por lo que solamente se combinó con la representación sin TF-IDF.

- Los experimentos que involucraron LDA fueron divididos en tres grupos, de acuerdo a los valores de alfa, tal como se explicó en la sección 3.7.5. Además, se puede observar la incorporación de combinaciones entre LDA y la distancia Coseno. Éstas consisten en experimentos adicionales donde se aprovechó el hecho de que LDA genera representaciones vectoriales sobre las cuales es posible

aplicar la función coseno.

- La tabla muestra los valores de la precisión, la sensibilidad y el puntaje F1, utilizando las modalidades macro ponderadas (como se definió en la sección 3.7.6). Nótese que existe redundancia entre el valor de la exactitud y la sensibilidad, debido a que son equivalentes cuando se utiliza el cálculo ponderado. Para efectos del análisis de las siguientes secciones, solamente se toman en cuenta la exactitud y el puntaje F1.

- Para cada conjunto de experimentos sobre el mismo conjunto de datos y mismo valor de k, se resaltaron los valores más altos obtenidos en cada una de las métricas.

En las siguientes secciones se presenta la prueba de hipótesis, se realiza la comparación de las distintas métricas de distancia de texto y se detalla el posterior análisis de resultados, incluyendo experimentos adicionales realizados para intentar entender el comportamiento observado.

4.1. Prueba de hipótesis

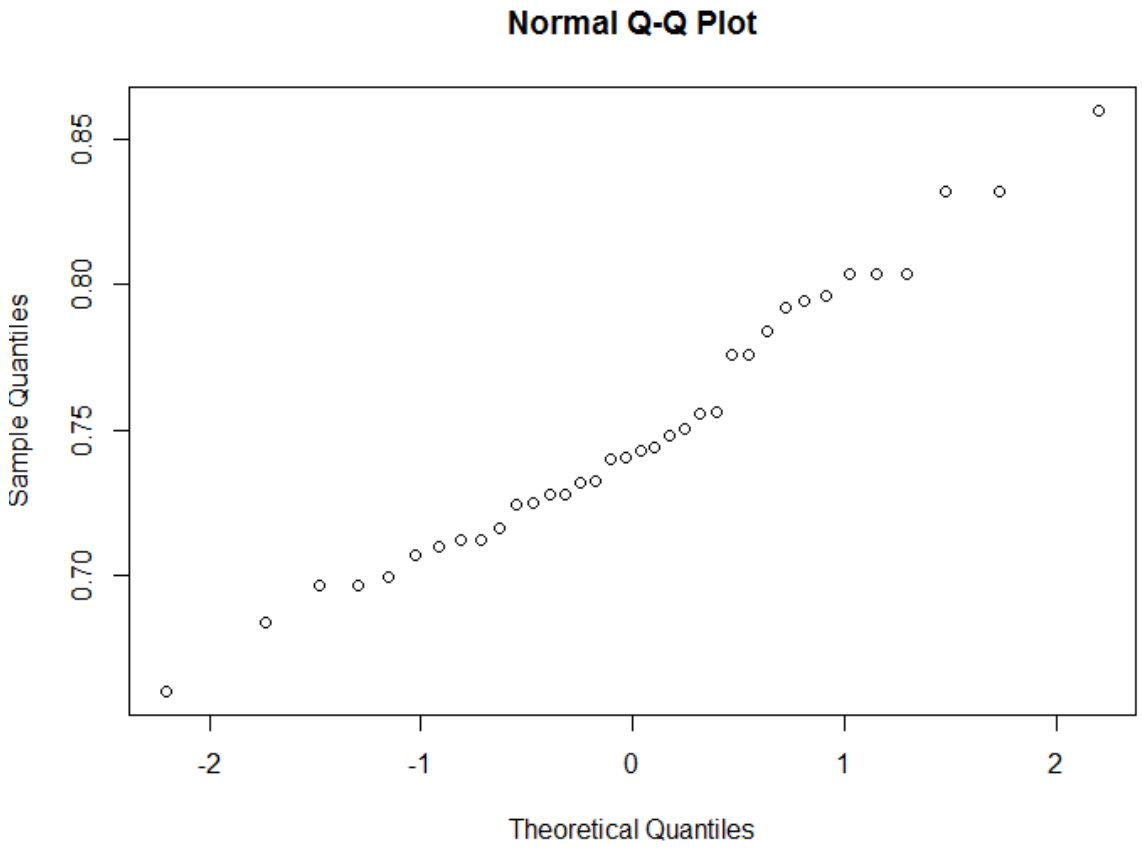
Se realizaron pruebas de normalidad sobre todos los valores de exactitud mostrados en la tabla 4.1. En la figura 4-1 se muestra el grafico Q-Q en el cual se nota una tendencia lineal indicando normalidad en los datos. Esto se reforzó con una prueba de Lillifors, la cual evalúa la hipótesis nula de que los valores están distribuidos normalmente, contra la hipótesis alternativa de que no lo están. El resultado de esta prueba se puede observar en la imagen 4-2, donde se obtuvo un valor p de 0.2864, el cual indica que los valores sí son normales considerando un nivel de significancia de 0.05 ($p > 0{,}05$).

Comprobada la normalidad de los datos, se procedió a realizar el análisis de la varianza, tomando como variable cualitativa la combinación representación-métrica y como variable cuantitativa la exactitud. El resultado obtenido utilizando la prueba

Tabla 4.1: Resultados obtenidos para cada experimento utilizando *bagging* por conjunto de datos, valor de k y combinación representación-métrica

Conjunto de datos	k	Representación / Metrica	Exactitud	Precisión	Sensibilidad	F1
reuters-1	1	LDA alfa=0.2 / Coseno	0.792	0.781184	0.792	0.777647
reuters-1	1	LDA alfa=0.5 / Coseno	0.796	0.766725	0.796	0.776012
reuters-1	1	LDA alfa=0.8 / Coseno	0.776	0.767779	0.776	0.766546
reuters-1	1	LDA alfa=0.2 / Divergencia KL	0.804	0.776947	0.804	0.772981
reuters-1	1	LDA alfa=0.5 / Divergencia KL	0.804	0.773775	0.804	0.778364
reuters-1	1	LDA alfa=0.8 / Divergencia KL	0.804	0.776947	0.804	0.772981
reuters-1	1	Vector de palabras con TF-IDF / Coseno	0.832	**0.840187**	0.832	0.822432
reuters-1	1	Vector de palabras sin TF-IDF / Coseno	**0.86**	0.840154	**0.86**	**0.844868**
reuters-1	1	Vector de palabras sin TF-IDF / Jaccard	0.832	0.80236	0.832	0.806325
reuters-1	11	LDA alfa=0.2 / Coseno	0.748	0.688388	0.748	0.70089
reuters-1	11	LDA alfa=0.5 / Coseno	0.74	0.665505	0.74	0.67611
reuters-1	11	LDA alfa=0.8 / Coseno	0.744	0.639617	0.744	0.674836
reuters-1	11	LDA alfa=0.2 / Divergencia KL	0.732	0.621309	0.732	0.652917
reuters-1	11	LDA alfa=0.5 / Divergencia KL	0.724	0.645322	0.724	0.661722
reuters-1	11	LDA alfa=0.8 / Divergencia KL	0.66	0.569169	0.66	0.570078
reuters-1	11	Vector de palabras con TF-IDF / Coseno	**0.776**	**0.703381**	**0.776**	**0.723759**
reuters-1	11	Vector de palabras sin TF-IDF / Coseno	0.756	0.664563	0.756	0.697859
reuters-1	11	Vector de palabras sin TF-IDF / Jaccard	0.716	0.609479	0.716	0.636426
reuters-2	1	LDA alfa=0.2 / Coseno	0.712082	0.648669	0.712082	0.672653
reuters-2	1	LDA alfa=0.5 / Coseno	0.727506	0.665563	0.727506	0.688994
reuters-2	1	LDA alfa=0.8 / Coseno	0.706941	0.670164	0.706941	0.67461
reuters-2	1	LDA alfa=0.2 / Divergencia KL	0.74036	0.703899	0.74036	0.711298
reuters-2	1	LDA alfa=0.5 / Divergencia KL	0.742931	0.675888	0.742931	0.69924
reuters-2	1	LDA alfa=0.8 / Divergencia KL	0.732648	0.701187	0.732648	0.70755
reuters-2	1	Vector de palabras con TF-IDF / Coseno	**0.794344**	**0.776115**	**0.794344**	**0.769744**
reuters-2	1	Vector de palabras sin TF-IDF / Coseno	0.755784	0.687079	0.755784	0.715314
reuters-2	1	Vector de palabras sin TF-IDF / Jaccard	0.784062	0.746229	0.784062	0.753398
reuters-2	11	LDA alfa=0.2 / Coseno	0.696658	0.548335	0.696658	0.597486
reuters-2	11	LDA alfa=0.5 / Coseno	0.696658	0.542021	0.696658	0.593143
reuters-2	11	LDA alfa=0.8 / Coseno	0.709512	0.552213	0.709512	0.617626
reuters-2	11	LDA alfa=0.2 / Divergencia KL	0.699229	0.556134	0.699229	0.611298
reuters-2	11	LDA alfa=0.5 / Divergencia KL	0.683805	0.517681	0.683805	0.571059
reuters-2	11	LDA alfa=0.8 / Divergencia KL	0.712082	**0.620116**	0.712082	0.626208
reuters-2	11	Vector de palabras con TF-IDF / Coseno	**0.750643**	0.618798	**0.750643**	**0.666292**
reuters-2	11	Vector de palabras sin TF-IDF / Coseno	0.724936	0.590299	0.724936	0.632797
reuters-2	11	Vector de palabras sin TF-IDF / Jaccard	0.727506	0.602923	0.727506	0.638196

Figura 4-1: Gráfico Q-Q de normalidad sobre los valores de exactitud obtenidos en los experimentos



de ANOVA se muestra en la figura 4-3.

Se observa que el valor p fue de 0.627, el cual, considerando un nivel de significancia del 0.05, permite aceptar la hipótesis nula: no existen diferencias significativas en las medias de las poblaciones determinadas por la representación de datos y la métrica.

Por lo tanto, estadísticamente, con una certeza de 95 %, no hay evidencia suficiente para determinar que los experimentos en los cuales se utilizó LDA y Divergencia KL obtuvieron mejor exactitud que las medidas base, al combinarlos mediante el algoritmo agregado *bagging*. Esto tampoco permite afirmar que la misma combinación obtiene peores resultados, ya que el análisis estadístico solamente sugiere que no hay diferencias significativas en los resultados respecto a las otras combinaciones. De esta

Figura 4-2: Resultado de la prueba Lillifors de normalidad sobre los valores de exactitud obtenidos en los experimentos

```
        Lilliefors (Kolmogorov-Smirnov) normality test

data:  cAccuracy
D = 0.11333, p-value = 0.2864
```

Figura 4-3: Resultado de ANOVA sobre los valores de exactitud obtenidos en los experimentos

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
representation_metric	8	0.01388	0.001735	0.777	0.627
Residuals	27	0.06034	0.002235		

manera, se rechaza la hipótesis de investigación.

Adicionalmente, se realizó el mismo análisis estadístico pero tomando en cuenta el puntaje F1 ponderado como variable cuantitativa. Sin embargo, se llegó a la misma conclusión (ver resultados en el apéndice E.1).

4.2. Análisis de resultados

El hecho de que el análisis de la varianza determinara que los resultados obtenidos en todos los experimentos son estadísticamente iguales, puede verse como algo positivo a destacar en esta investigación por varias razones. En primer lugar, se está evaluando una representación de documentos y una métrica cuya combinación no ha sido evaluada en ninguna otra investigación como solución a la clasificación de texto para imputar valores faltantes, y que, sin embargo, obtuvo resultados similares a técnicas ampliamente utilizadas. Por otro lado, LDA logra una enorme reducción dimensional como método de representación de texto, al mismo tiempo que obtiene resultados comparables a los experimentos que utilizaron vector de palabras. Por último, el hecho de que las diferencias en la exactitud y el F1 obtenidas en los experimentos no sean significativas, sirve de motivación para continuar buscando áreas

de mejora en las técnicas de modelado de temas y distribuciones de probabilidad, de modo que lleguen a superar los métodos tradicionales.

Si se analiza el desempeño general de las técnicas utilizadas, los resultados de exactitud mostrados en la tabla 4.1 se pueden considerar bajos para todos los experimentos. Esto porque los puntajes estuvieron por debajo de 0.86 y en la mayoría de los casos la exactitud fue menor a 0.8. Se puede observar una situación muy similar con respecto al puntaje F1 ponderado.

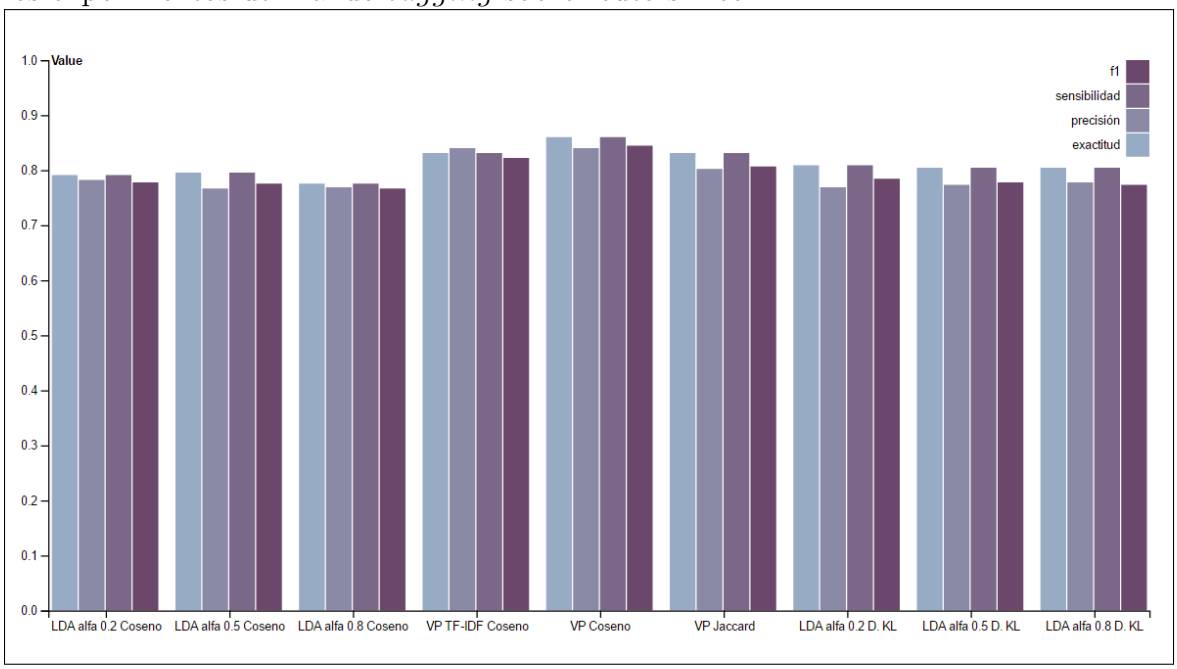
Además, aunque la prueba estadística no logró demostrar si alguna combinación supera a las demás, se evidencia que la combinación de vector de palabras usando TF-IDF con la métrica Coseno logró mejores puntos de exactitud, precisión, sensibilidad y puntaje F1 en todos los experimentos (valores resaltados en la tabla 4.1). También se puede observar que los resultados puntuales de los experimentos con LDA fueron los más bajos en la mayoría de los casos, principalmente cuando se utilizó la función coseno como métrica de distancia de texto. Estas comparaciones se pueden apreciar de manera más clara en las figuras 4-4 y 4-5, donde los tres grupos de barras para los puntajes obtenidos usando vector de palabras (etiquetados como VP TF-IDF Coseno, VP Coseno y VP Jaccard), están ligeramente por encima de los resultados obtenidos usando LDA, tanto para reuters-1 como para reuters-2.

En las siguientes secciones se analizan los puntos negativos mencionados anteriormente y propuestas que permiten mejorar los resultados como potencial trabajo futuro.

4.2.1. Evaluación de mejoras para aumentar los valores generales de exactitud y puntaje F1

Aunque los grados de exactitud y precisión aceptables dependen mucho del problema a resolver, se lograron encontrar mejoras a los experimentos que permitieron

Figura 4-4: Gráfico. Puntajes de exactitud, precisión, sensibilidad y F1 obtenidos en los experimentos utilizando *bagging* sobre reuters-1 con k=1.



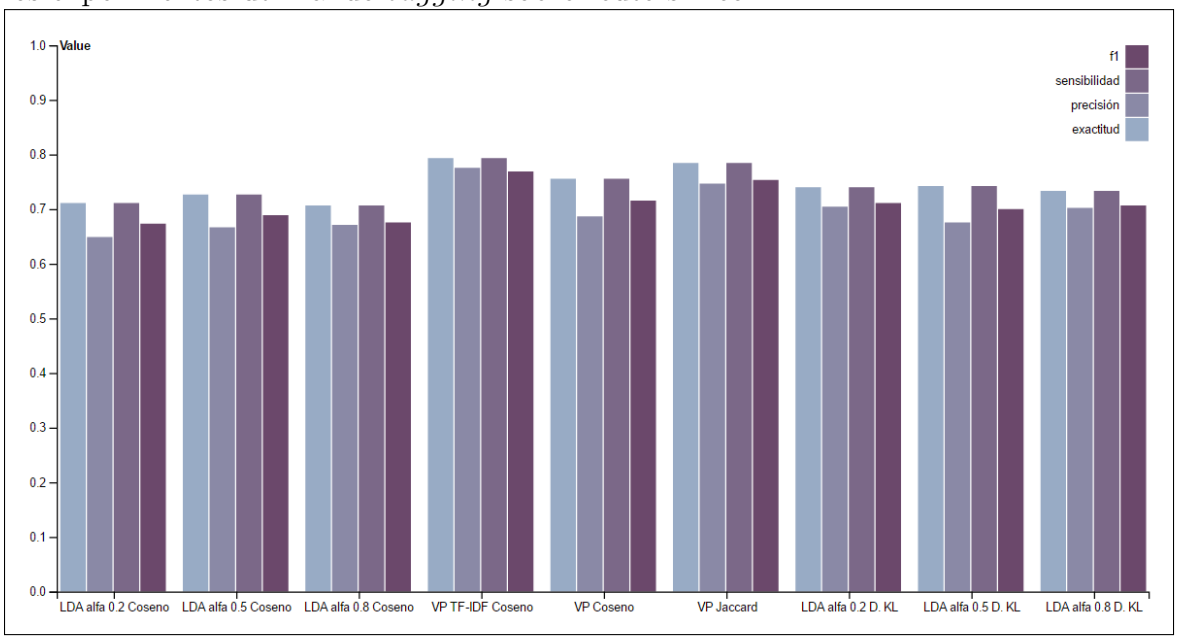
obtener resultados más favorables.

Los tres factores evaluados para mejorar los resultados fueron: 1) el ajuste del tamaño de las muestras utilizadas por el algoritmo de KNN con *bagging*, 2) el uso de KNN sin método agregado (KNN simple) y 3) la distribución uniforme de las clases en los conjuntos de datos. Cada uno de ellos se detalla en las siguientes secciones.

Tamaño de las muestras en el algoritmo *bagging*

Uno de los parámetros que podría haber contribuido a la obtención de resultados no muy favorables, fue el tamaño de las muestras a utilizar por el método agregado. En la sección 3.7.5 se explicó la razón del por qué dicho valor se fijó en 0.15 para todos los experimentos. Sin embargo, para el análisis se decidió ejecutar una ronda de experimentos incrementado dicho valor a 0.25 (lo que significó que cada conjunto de datos era dividido en cuatro grupo por el algoritmo *bagging*). Aunque los resultados obtenidos usando 0.25 superaron en su mayoría a los resultados iniciales, la diferencia no es mucha y los puntajes de exactitud, precisión, sensibilidad y F1 siguen siendo

Figura 4-5: Gráfico. Puntajes de exactitud, precisión, sensibilidad y F1 obtenidos en los experimentos utilizando *bagging* sobre reuters-2 con k=1.



menores a 0.9. Se intentó recolectar más resultados incrementando este parámetro, sin embargo, la cantidad de recursos requeridos era muy alto y provocó fallos en el programa debido a memoria insuficiente.

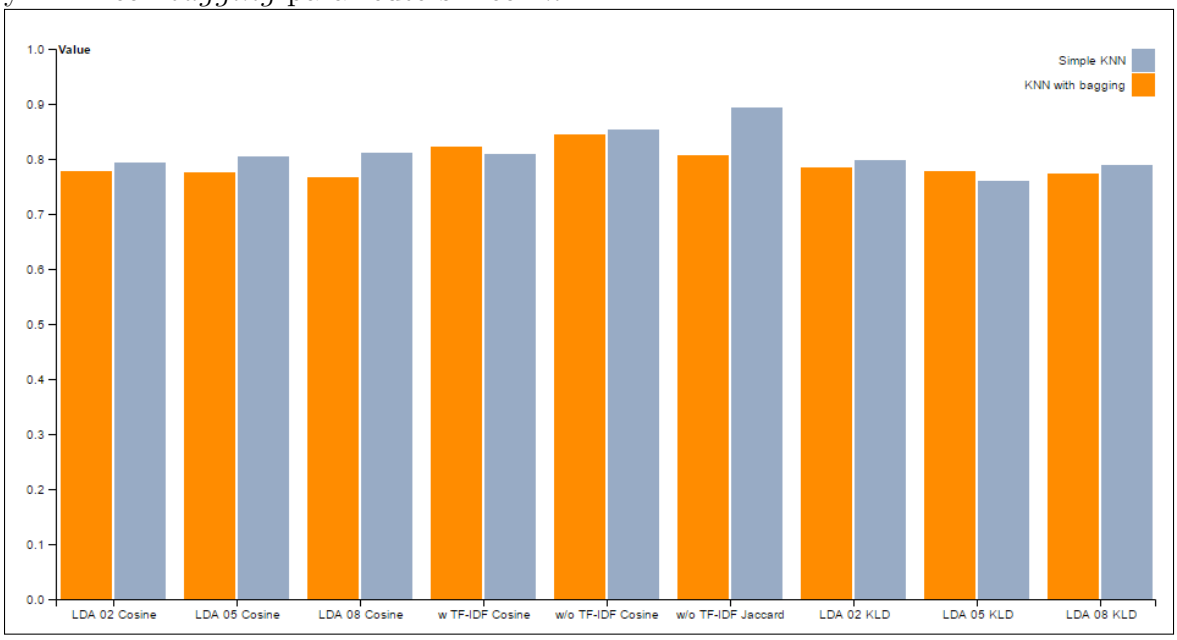
Resultados utilizando KNN sin método agregado

Se ejecutó una ronda completa de experimentos utilizando KNN simple, sin ningún método agregado, y bajo las mismas condiciones de los experimentos originales. En este caso, se obtuvieron los resultados que se muestran en la tabla del apéndice D.1. Se nota un incremento considerable en los puntajes, al observar valores de exactitud y F1 muy cercanos a 0.9 para reuters-1 con $k = 1$, además de que en la mayoría de experimentos los puntajes son mayores a 0.8. En la tabla se resaltan los valores máximos obtenidos para cada variable y cada combinación conjunto de datos-valor de k, donde sobresale Jaccard con vector de palabras sin TF-IDF.

Para efectos de comparación, se realizó un análisis estadístico sobre los valores de exactitud y F1 considerando el algoritmo (*bagging* o KNN simple) como variable

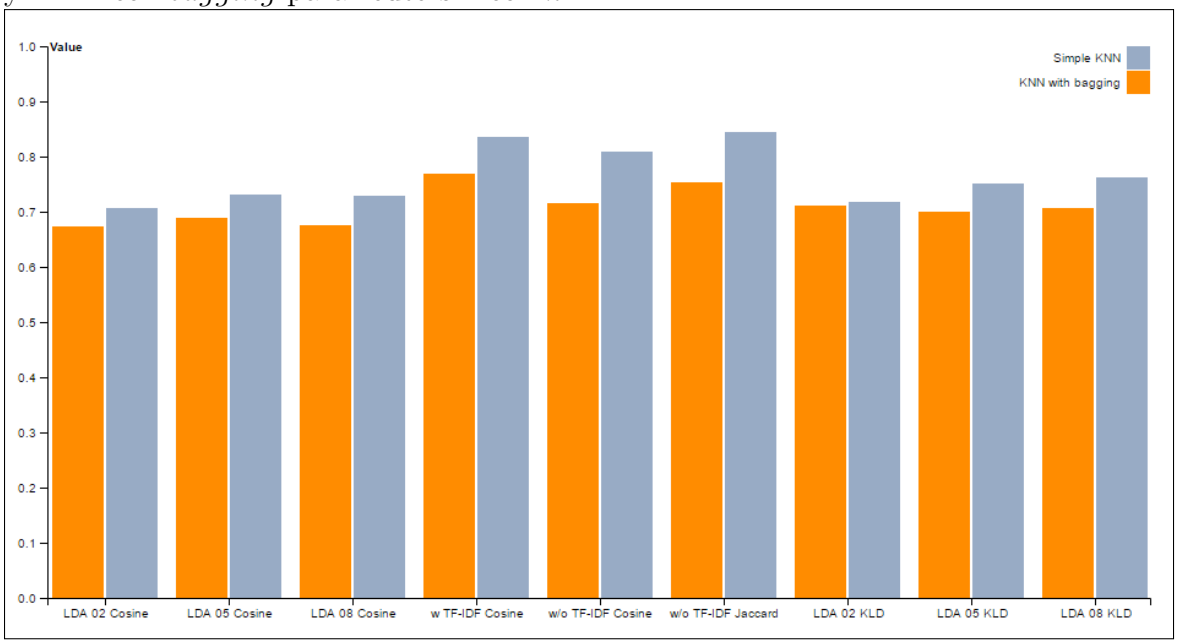
cuantitativa (ver apéndice E.3). Tanto para exactitud como para F1, se encuentran diferencias significativas al aplicar la prueba de ANOVA (observándose más diferencia para F1). En los gráficos de las figuras 4-6 y 4-7 se comparan los puntajes de F1 ponderado obtenidos usando KNN simple y KNN con *bagging* para reuters-1 y reuters-2 respectivamente. KNN simple supera a *bagging* en la mayoría de los casos y la diferencia se hace más evidente en los resultados para el conjunto de datos reuters-2.

Figura 4-6: Gráfico. Comparación de puntajes de F1 obtenidos usando KNN simple y KNN con *bagging* para reuters-1 con $k = 1$.



La razón del por qué se da esta diferencia y mejoría en los resultados al no utilizar un método agregado, se puede explicar basándose en el propósito para el cual *bagging* es utilizado en clasificación. El método *bagging* permite disminuir los errores de varianza que pueden llegar a existir en un modelo de clasificación, pero no mejora errores de sesgo [2]. Esto significa que *bagging* ayuda a que las imputaciones de datos sean más consistentes, si se utilizaran diferentes conjuntos de datos de entrenamiento bajos las mismas condiciones, pero la exactitud no mostraría mejora si existen errores de sesgo (por ejemplo, baja calidad en los datos). Por lo tanto, estos resultados sugieren que los modelos de clasificación generados en esta investigación no presentaron

Figura 4-7: Gráfico. Comparación de puntajes de F1 obtenidos usando KNN simple y KNN con *bagging* para reuters-2 con $k = 1$.



altos errores de varianza, por lo que la utilización de *bagging* no contribuyó a mejorar el algoritmo KNN, sino que más bien pudo contribuir a aumentar el error de sesgo en cada experimento.

Viéndolo de otra manera, basándose en el concepto de diversidad descrito en la sección 2.5, se podría concluir que los clasificadores individuales creados con *bagging* no fueron los suficientemente diversos para justificar la utilización de un método agregado.

Conjuntos de datos con clases balanceadas

Dado que la calidad de los datos es un factor fundamental en el proceso de clasificación y aprendizaje supervisado, se quiso evaluar el caso hipotético de que las etiquetas en los conjuntos de datos estuvieran distribuidos de forma equitativa. Para esto, se creó un nuevo conjunto de datos tomado de reuters-1, procurando que el número de elementos bajo una clase fuera igual o similar al de las demás.

El conjunto de datos reuters-1 modificado consistió en 104 elementos distribuidos entre 4 clases de aproximadamente 25 elementos cada una. Los resultados obtenidos, luego de ejecutar todos los experimentos para $k = 1$, mostraron un incremento significativo en los valores de exactitud y F1 en la mayoría de los casos. Por ejemplo, varios valores de exactitud estuvieron por encima de 0.9, siendo Jaccard con KNN simple el método que obtuvo el mejor resultado con 0.96 de exactitud.

Importante aclarar que estos resultados solamente ayudan a respaldar la idea de que la calidad de los conjuntos de datos afectó considerablemente los resultados principales. En problemas reales es muy improbable que se encuentren conjuntos de datos uniformemente etiquetados e incluso, 104 elementos es un número muy bajo para considerar resolver este tipo de problema con algoritmos de aprendizaje máquina. Se pueden encontrar más detalles sobre los resultados de este experimento en el apéndice D.2, donde también se evidencian mejores resultados al no utilizar *bagging*.

4.2.2. Análisis de resultados obtenidos en los experimentos que involucraron LDA

Dados los resultados obtenidos en la prueba de hipótesis, además de un análisis de la varianza adicional realizado incluyendo los resultados al utilizar KNN simple (ver detalles en apéndice E.5), se demuestra que LDA en general no logró superar la exactitud ni el F1 obtenidos usando métodos convencionales. En esta sección se procede a analizar la razón del por qué dichas técnicas propuestas mostraron un bajo desempeño, haciendo uso de los datos obtenidos a lo largo de la investigación. Se avaluaron varios escenarios en los cuales LDA y la divergencia KL pueden llegar a superar los métodos tradicionales. Para el análisis de esta sección, se consideran los resultados obtenidos al usar, principalmente, KNN simple y el conjunto de datos reuters-1, los cuales muestran mayor exactitud tal como se describió en las secciones anteriores.

Desempeño de la Divergencia KL como métrica de distancia de texto

Se buscó comprobar si el hecho de haber utilizado la divergencia KL como métrica de distancia de texto, influyó en los resultados negativos de los experimentos con LDA. Por esta razón, se ejecutó el mismo conjunto de experimentos pero utilizando la distancia coseno como métrica para comparar vectores de temas. Estos resultados ya están incluidos en la tabla 4.1 y en el apéndice D.1.

En los resultados de la tabla 4.1, no se observan diferencias significativas entre los valores obtenidos utilizando coseno y los obtenidos usando divergencia KL. De igual forma, en la tabla del apéndice D.1 se aprecian ciertas diferencias en los puntajes de exactitud y F1 obtenidos al comparar ambas métricas, pero tampoco hay suficiente evidencia para determinar si una es mejor que la otra. Algo importante a destacar, es el hecho de que en ambas tablas, para la mayoría de los experimentos, LDA obtuvo mejores resultados al utilizar la divergencia KL. Esto tiene sentido dado que dicha métrica funciona con distribuciones de probabilidad, no así la distancia coseno.

Tamaño de los documentos de texto

Se buscó analizar un experimento que involucrara LDA y la divergencia KL, con el fin de encontrar algún patrón en los datos que fueron imputados correctamente y en aquellos donde la imputación no fue la correcta. Específicamente se analizó el experimento para KNN simple con $k = 1$, cuyos resultados de imputación para cada uno de los documentos se muestra en las tablas del apéndice D.3. Un aspecto interesante observado fue el tamaño en bytes de los documentos de texto, donde el tamaño promedio de los textos fue de 635 bytes para aquellos clasificados correctamente y 963 bytes para los documentos imputados de manera incorrecta. En los datos del apéndice D.3 se observa cómo los documentos de 900 bytes o más, solamente representaron un 20 % de los textos imputados correctamente, mientras que dicho tamaño representó casi el 50 % de los documentos clasificados incorrectamente.

Basándose en estas observaciones, se creó un conjunto de datos adicional formado únicamente por las representaciones de los textos cuyos tamaños fueran menores a los 150 bytes, tomados de reuters-1. Este conjunto de datos de textos cortos consistió en 129 documentos para entrenamiento y 34 documentos para pruebas. Se ejecutaron experimentos para todas las combinaciones estudiadas con k=1, dando como resultado los valores que se adjuntan en el apéndice D.4. Para *bagging* y para KNN simple, las combinaciones LDA-Divergencia KL y LDA-Coseno obtuvieron los mejores resultados de exactitud, respectivamente, y por encima de 0.8, tal como se resalta en la tabla.

Similar al ejemplo de la sección 4.2.1, el tamaño y la manera en la que se crearon estos conjuntos de datos, así como el número de experimentos realizados, no aportan suficiente evidencia para garantizar la efectividad de una combinación sobre las demás bajo estas circunstancias. Sin embargo, estos resultados preliminares abren la puerta a futuras investigaciones que permitan demostrar si, en efecto, el modelado de temas mediante LDA podría llegar a desempeñarse mejor que los métodos tradicionales cuando se trata de textos cortos.

Optimización del modelo de LDA mediante el número de temas

Tanto la función de similitud de texto como la calidad de los datos descritos en las secciones anteriores, son factores que influyeron en los resultados finales, tal como sugieren los resultados expuestos. Existe otro factor determinante: el modelo de temas generado por el algoritmo de LDA supervisado. Este modelo debió ser capaz de abstraer, de manera eficiente, la representación de todos los documentos de texto en un número reducido de dimensiones (en comparación con los vectores de palabras). La calidad de estas representaciones es fundamental para que KNN y las métricas de distancia logren clasificar el texto de manera similar a como lo harían con vectores de palabras.

Se buscó optimizar este modelo mediante el parámetro número de temas, que inicialmente fue establecido de acuerdo al número de clases en cada conjunto de datos,

tal como se mencionó en la sección 3.7.5. En LDA supervisado, el número de temas no está necesariamente ligado al número de clases diferentes, ya que un documento etiquetado bajo cierta clase puede llegar a tener muchos temas que no se relacionan con las otras clases. Se utilizó como base el conjunto de datos reuters-1 y se generaron dos modelos adicionales de LDA, uno para 100 temas y otro para 200 temas, y se ejecutaron experimentos para k=1 con métricas de coseno y divergencia KL. Los resultados obtenidos se pueden apreciar en la tabla del apéndice D.5, donde se comparan los valores de exactitud y F1 para los tres modelos, usando valores de alfa 0.2 y 0.8.

Los resultados de los experimentos aplicados sobre los modelos de LDA con alfa 0.2, parecen incrementar en cuanto a exactitud y F1 conforme aumenta el número de temas, como se observa en los gráficos de las figuras 4-8 y 4-9. Esta tendencia se aprecia principalmente al utilizar la divergencia KL, no así para los experimentos con alfa=0.8. Esto sugiere que, conforme se aumenta el número de dimensiones, LDA y la divergencia KL pueden llegar a obtener mejores resultados, siempre y cuando el número de temas por documento sea bajo (valor de alfa cercano a cero en la función Dirichlet). Esto tiene sentido, dado que un incremento en el número de dimensiones disminuye el nivel de abstracción de los datos (logrando igual una enorme reducción dimensional comparado con el vector de palabras) y un valor bajo de alfa permite darle más peso a pocos temas que pueden llegar a representar mejor cada documento. Aunque este enfoque suena prometedor, el costo computacional que involucra agregar más temas es muy grande, principalmente en modelos como LDA donde se trabaja con heurísticas y técnicas de *Expectation-Maximization* que deben ser capaces de lograr la convergencia de un modelo complejo. Aunque la medición del tiempo de ejecución no estuvo dentro del enfoque de este proyecto, se observó que el tiempo requerido para la obtención de los modelos de 100 y 200 temas estuvo muy por encima de los demás algoritmos (Por ejemplo, para 200 temas se tardó casi un día completo para generar cada modelo LDA, en comparación con los vectores de palabras cuyo proceso de generación y almacenamiento tomó alrededor de 3 horas).

Figura 4-8: Gráfico. Tendencia de los valores de exactitud al aplicar KNN simple y KNN con *bagging* con $k = 1$ para diferentes números de temas y $\alpha = 0{,}2$ con LDA.

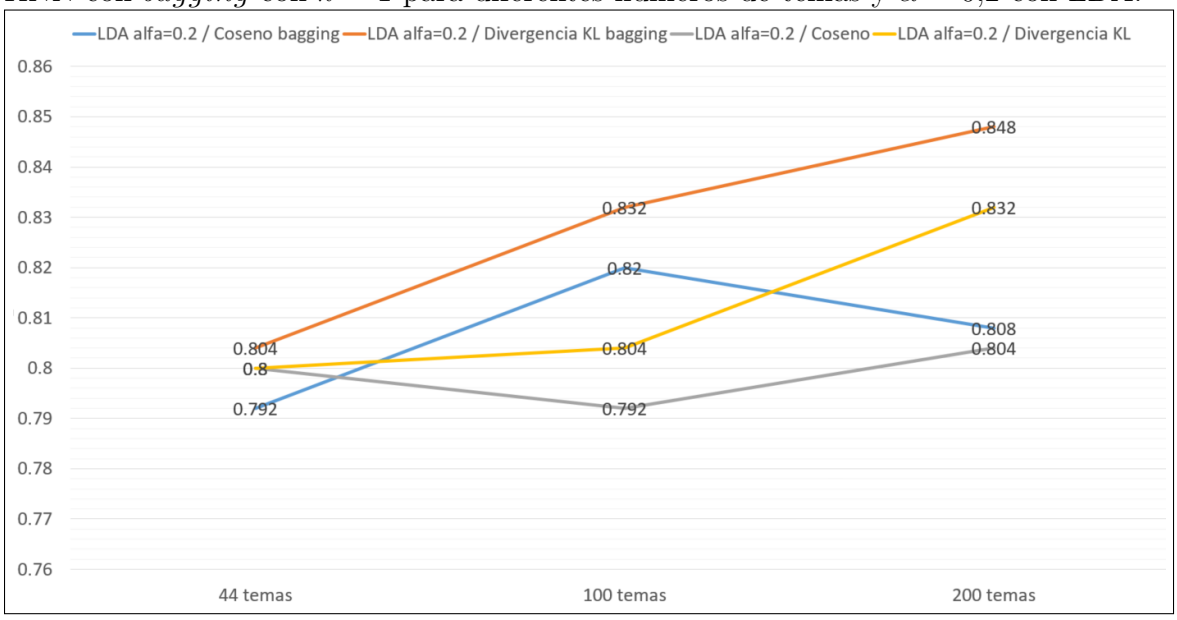
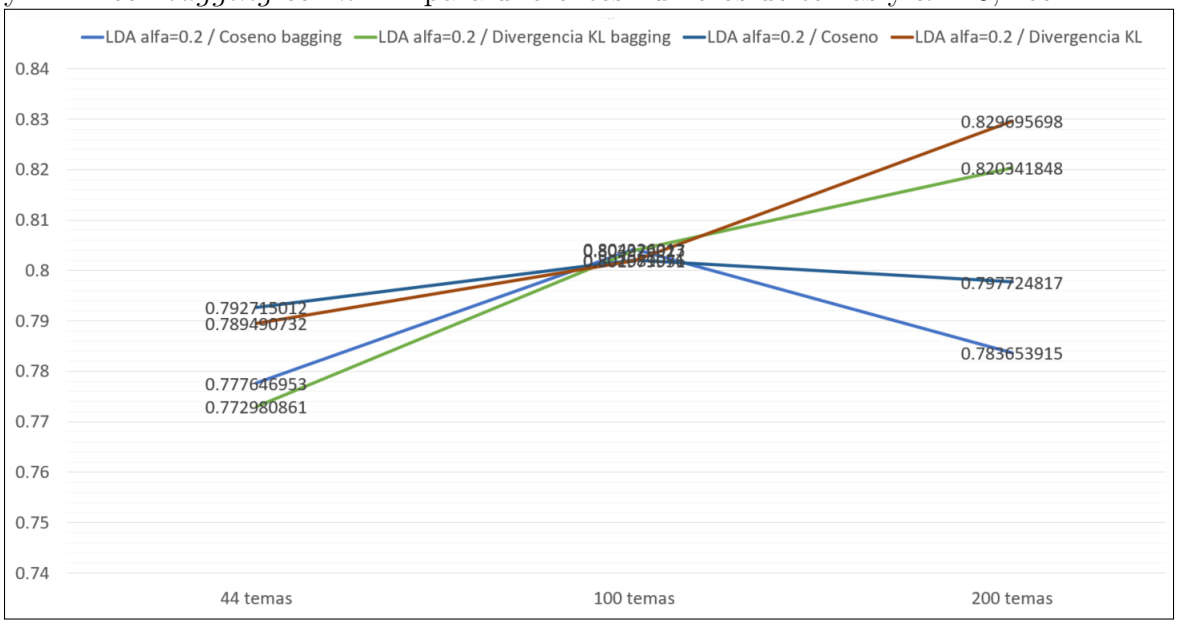


Figura 4-9: Gráfico. Tendencia de los puntajes de F1 ponderado al aplicar KNN simple y KNN con *bagging* con $k = 1$ para diferentes números de temas y $\alpha = 0{,}2$ con LDA.



Capítulo 5

Conclusiones y Trabajo Futuro

En este capítulo se resumen los aspectos claves del trabajo realizado, así como los principales hallazgos y potencial trabajo futuro para complementar esta investigación.

5.1. Conclusiones

En este proyecto se realizó un estudio sobre diferentes métricas de distancia de texto, aplicadas a la clasificación de documentos para imputación de valores faltantes. Estas métricas involucran métodos de representación de documentos de texto, así como funciones matemáticas aplicadas sobre dichas representaciones. Se evaluó, principalmente, la exactitud obtenida al momento de imputar valores categóricos faltantes utilizando el algoritmo KNN y el método agregado *bagging*. El desarrollo de la investigación consistió en lo siguiente:

- Se prepararon dos conjuntos de datos, cada uno con registros de dos atributos: uno de tipo texto y otro de tipo categórico.
- Los conjuntos de datos fueron transformados a distintas representaciones vectoriales, mediante programas y bibliotecas diseñados para trabajar con documentos de texto. Estas representaciones consistieron en vectores de frecuencias de palabras, vectores de pesos TF-IDF y vectores de probabilidades de temas (haciendo uso del algoritmo LDA supervisado para diferentes valores de alfa).

- Se implementó un programa capaz de aplicar el algoritmo KNN y el método agregado *bagging* sobre las representaciones de datos, utilizando tres métricas de distancia de texto: Coseno, Jaccard y divergencia Kullback Leibler.

- Utilizando las representaciones de texto y el programa implementado, se ejecutaron varios experimentos que combinaban las representaciones de datos con las métricas de distancia de texto. Esto mediante un conjunto de datos de entrenamiento y un conjunto de datos de prueba, capturando los valores imputados en cada experimento.

- Luego de ejecutar los experimentos, se calcularon medidas de desempeño para cada uno, principalmente la exactitud y el puntaje F1 ponderado.

- Se realizó una prueba de hipótesis con los resultados de exactitud obtenidos. La hipótesis se rechaza debido a que, estadísticamente, no hay diferencias entre los resultados obtenidos utilizando LDA y divergencia KL en comparación con los otros métodos, por lo tanto no se puede determinar si el desempeño de dicha combinación es mejor que los demás.

- Por último, se analizaron los resultados obtenidos con el fin de entender el comportamiento observado en los experimentos.

A pesar de que el análisis estadístico rechazó la hipótesis de investigación, se obtuvieron importantes conclusiones basándose en los resultados obtenidos en los experimentos principales y en las pruebas ejecutadas adicionalmente como parte del análisis. Los principales hallazgos se listan a continuación:

- En general, se observó que el método agregado *bagging* no logró mejorar la exactitud del algoritmo KNN, sino que por el contrario obtuvo un menor desempeño en la mayoría de experimentos. Esto permitió demostrar que, bajo los escenarios estudiados en esta investigación, los clasificadores KNN entrenados utilizando pequeñas muestras del conjunto original, no fueron lo suficientemente variados ni representaron altos errores de varianza. Estos aspectos deben ser considerados al momento de decidir si es conveniente o no utilizar un método agregado.

- La calidad de los conjuntos de datos influyó en la exactitud y el puntaje F1 obtenidos al momento de imputar los valores faltantes en todos los experimentos. Por ejemplo, tanto en reuters-1 como en reuters-2, existía una distribución desbalanceada de clases, donde una única clase abarcaba casi el 50 % de los datos. Se observó un incremento significativo en el valor de la exactitud general, al ejecutar un experimento utilizando un conjunto de datos hipotético con clases balanceadas. Para efectos prácticos, el tratar con documentos etiquetados de forma variada resulta un reto interesante a explorar como trabajo futuro.

- LDA logra una importante reducción dimensional como método de representación de texto y, al combinarlo con la divergencia KL, logró resultados comparables a los experimentos que utilizaron vectores de palabras con coseno y Jaccard. Para ambos conjuntos de datos, el número de dimensiones en los vectores de temas fue 150 veces más pequeño que en los vectores de palabras, logrando abstraer miles de palabras en menos de 60 temas para los experimentos principales.

- Varios de los resultados obtenidos sugieren que LDA y la divergencia KL pueden lograr un mejor desempeño cuando los datos contienen textos cortos, especialmente cuando se establece un número pequeño de temas al calcular el modelo. Para el caso de documentos de tamaños variados, como los utilizados en esta investigación, se observó que al aumentar el número de temas y mantener un valor de alfa bajo en los parámetros del modelado, incrementan la exactitud obtenida, al mismo tiempo que se mantiene la reducción de dimensiones.

- El mayor problema observado al utilizar modelado de temas como técnica de representación de documentos, fue el costo computacional y el tiempo de ejecución requeridos para calcular el modelo. Aunque dichas variables de respuesta no formaron parte del alcance de esta investigación, cabe destacar que el tiempo de ejecución requerido para formar los vectores de temas fue bastante alto (alrededor de un día completo de ejecución continua), especialmente si se compara con el tiempo requerido para formar un vector de palabras (menos de un minuto para generar el modelo). Este costo aumenta conforme más dimensiones y más

iteraciones de convergencia se establezcan.

Muchos de los aspectos estudiados a lo largo de esta investigación tienen potencial para ser explorados más a fondo o de maneras distintas, de forma que se mejoren o se complementen los resultados obtenidos. Estos aspectos se proponen en la siguiente sección.

5.2. Trabajo futuro

Este trabajo se enfocó en el análisis de diferentes algoritmos con la idea de que los mismos pudieran ser aplicados de forma general, independiente del conjunto de datos. Sin embargo, como se observó en los resultados de esta investigación y de otros trabajos similares, la solución a este tipo de problemas depende mucho del conjunto de datos específico sobre el cual se quiere trabajar. En este sentido, se podría trabajar con un mayor número de conjuntos de datos diferentes, y analizar los resultados basándose en las características específicas de cada uno. Un gran aporte consistiría en aplicar estas técnicas a bases de datos empresariales que cumplan las condiciones que motivaron a realizar esta investigación. Esto permitiría entender mejor el efecto causado al aplicarse sobre datos reales y ayudaría tener una mejor visibilidad sobre las aplicaciones prácticas de estos estudios a nivel empresarial.

Con respecto al preprocesamiento de los datos, resultaría interesante evaluar diferentes implementaciones para una misma técnica y determinar si existen diferencias en cuanto a desempeño. Por ejemplo, evaluar si todos los paquetes de R o de Python se comportan igual al momento de eliminar palabras no relevantes, al hacer derivación de palabras o al ajustar pesos mediante TF-IDF. Dada la importancia que tuvo la representación de datos en esta investigación, un trabajo de este tipo podría causar gran influencia en el resultado final de clasificación.

Como se mencionó en varias ocasiones a lo largo de este documento, el alcance, en cuanto a las representaciones y medidas de similitud de texto, se limitó a las técnicas

estadísticas. Estas técnicas se enfocan más que todo en el análisis léxico y aún poseen muchas áreas por explorar, tal como el modelado de temas estudiado en este proyecto. Sin embargo, un potencial aporte a los resultados presentados, sería la incorporación de técnicas de análisis semántico [36, 28, 20], de manera que se puedan comparar con los métodos estudiados.

El algoritmo de clasificación KNN y el método agregado *bagging* permitieron facilidad y flexibilidad en el diseño de experimentos, principalmente desde el punto de vista técnico al momento de implementar los programas. Como trabajo futuro, se podría considerar el uso de otros algoritmos de clasificación supervisado. Esto, además de aportar diferentes soluciones al problema, podría lograr más variedad en los clasificadores, de forma que se aprovechen las propiedades de *bagging*, e incluso se incorporen otros métodos agregados.

Para el caso específico del método LDA, existen muchos aspectos que no estuvieron dentro del alcance de este proyecto pero que podrían representar aportes valiosos a la investigación. En primer lugar, está el hecho de que únicamente se utilizaron las representaciones de documentos generadas por el modelo pero no se aprovechó el algoritmo completo de LDA supervisado como tal. El estudio y la implementación de este algoritmo, el cual no involucra KNN ni métricas de distancia de texto, podría aportar mejoras a los resultados, aprovechando todas las propiedades probabilísticas de la función Dirichlet y de sus métodos de optimización. En segundo lugar, muchos de los parámetros requeridos por LDA para la generación de los modelos, podrían estudiarse más a fondo, de manera que se logre un balance adecuado entre la exactitud del modelo y el tiempo requerido para construirlo. Esto conlleva al tercer aspecto, que al mismo tiempo representaría un gran reto, el cual sería la exploración de optimizaciones a las implementaciones existentes de LDA supervisado o, por otro lado, la búsqueda de métodos alternativos de modelado de temas, que sean más eficientes en cuanto a tiempo de ejecución y complejidad.

Finalmente, se podrían evaluar los tiempos de ejecución para cada uno de los experimentos, desde el momento en que los documentos son transformados hasta que se dé la imputación de la categoría faltante. En el caso de esta tesis, el proceso estuvo segmentado en diferentes programas implementados usando distintos lenguajes de programación, lo cual dificultó la medición de los tiempos. Aunque incorporar esta variable agrega más complejidad, la misma sería un gran aporte, pues permitiría identificar de manera más clara qué se compensa y qué se sacrifica (en términos de desempeño y de eficiencia) al utilizar una métrica de distancia de texto con respecto a otra.

Bibliografía

[1] Hervé Abdi and Lynne J Williams. Tukey's honestly significant difference (hsd) test. *Encyclopedia of Research Design. Thousand Oaks, CA: Sage*, pages 1–5, 2010.

[2] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[3] Christos Anagnostopoulos and Peter Triantafillou. Scaling out big data missing value imputations: pythia vs. godzilla. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 651–660. ACM, 2014.

[4] Jun Araki and Jamie Callan. An annotation similarity model in passage ranking for historical fact validation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1111–1114. ACM, 2014.

[5] Duck-Ho Bae, Seok-Ho Yoon, Tae-Hwan Eom, Jiwoon Ha, Young-Sup Hwang, and Sang-Wook Kim. Computing paper similarity based on latent dirichlet allocation. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, page 77. ACM, 2014.

[6] Douglas Bates. Fitting linear mixed models in r. *R news*, 5(1):27–30, 2005.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[8] Konstantinos Bougiatiotis and Theodoros Giannakopoulos. Content representation and similarity of movies based on topic extraction from subtitles. In *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, page 17. ACM, 2016.

[9] Wang Chong, David Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1903–1910. IEEE, 2009.

[10] Owen Davison, Abidalrahman Mohammad, and Evangelos E Milios. P-gtm: privacy-preserving google tri-gram method for semantic text similarity. In *Proceedings of the 2014 ACM symposium on Document engineering*, pages 81–84. ACM, 2014.

[11] Nikhil Devraj and Michael Chary. How do twitter, wikipedia, and harrison's principles of medicine describe heart attacks? In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 610–614. ACM, 2015.

[12] Dipankar Dutta and Paramartha Dutta. A real coded moga for mining classification rules with missing attribute values. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, pages 355–358. ACM, 2011.

[13] Bela A Frigyik, Amol Kapila, and Maya R Gupta. Introduction to the dirichlet distribution and related processes. department of electrical engineering, university of washignton. Technical report, UWEETR-2010-0006, 2010.

[14] Jun Gong, Lidan Wang, and Douglas W Oard. Matching person names through name transformation. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1875–1878. ACM, 2009.

[15] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[16] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[17] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[18] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[19] Tsunenori Ishioka. Investigations into missing values imputation using random forests for semi-supervised data. In *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services*, pages 296–301. ACM, 2014.

[20] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10, 2008.

[21] Timothy P Jurka, Loren Collingwood, Amber Boydstun, Emiliano Grossman, and Wouter van Atteveldt. Rtexttools: Automatic text classification via supervised learning. *R package version*, 1(9), 2012.

[22] Asjad M Khan, Kathryn S Mckinley, Rotem Bentzur, Daniel Feinberg, Daniel Frampton, Samuel Z Guyer, Martin Hirzel, Antony Hosking, Maria Jump, Han Lee, et al. A comparison of personal name matching: Techniques and practical issues. In *in 'Workshop on Mining Complex Data'(MCD'06), held at IEEE ICDM'06, Hong Kong*, 2006.

[23] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[24] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM, 2004.

[25] Alina Lazar, Sarah Ritchey, and Bonita Sharif. Improving the accuracy of duplicate bug report detection using textual similarity measures. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 308–311. ACM, 2014.

[26] Noah Liebman and Darren Gergle. Capturing turn-by-turn lexical similarity in text-based communication. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, pages 553–559. ACM, 2016.

[27] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

[28] Jorge Martinez-Gil and Mario Pichler. Analysis of word co-occurrence in human literature for supporting semantic correspondence discovery. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*, page 1. ACM, 2014.

[29] Ajay Singh Mavai and Sadhna K Mishra. A survey and comparative study of different data mining techniques to implement a missing value estimator system. 2014.

[30] Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[31] Donald Metzler, Susan Dumais, and Christopher Meek. *Similarity measures for short segments of text*. Springer, 2007.

[32] Douglas C Montgomery. *Design and analysis of experiments*. John Wiley & Sons, 2008.

[33] Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximisation techniques. *arXiv preprint arXiv:0704.3474*, 2007.

[34] Peter Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

[35] Maryam Sabzevari. *Ensemble Learning in the Presence of Noise*. PhD thesis, Universidad Autónoma de Madrid, 2015.

[36] Sepideh Seifzadeh, Ahmed K Farahat, Mohamed S Kamel, and Fakhri Karray. Short-text clustering using statistical semantics. In *Proceedings of the 24th International Conference on World Wide Web*, pages 805–810. ACM, 2015.

[37] Axel J Soto, Abidalrahman Mohammad, Andrew Albert, Aminul Islam, Evangelos Milios, Michael Doyle, Rosane Minghim, and Maria Cristina Ferreira de Oliveira. Similarity-based support for text reuse in technical writing. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 97–106. ACM, 2015.

[38] Ashok N Srivastava and Mehran Sahami. *Text mining: Classification, clustering, and applications*. CRC Press, 2009.

[39] Maria Terzi, Matthew Rowe, Maria-Angela Ferrario, and Jon Whittle. Text-based user-knn: Measuring user similarity based on text reviews. In *User Modeling, Adaptation, and Personalization*, pages 195–206. Springer, 2014.

[40] Cao Truong Tran, Mengjie Zhang, and Peter Andreae. Multiple imputation for missing data using genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 583–590. ACM, 2015.

[41] Pucktada Treeratpituk and C Lee Giles. Name-ethnicity classification and ethnicity-sensitive name matching. In *AAAI*, 2012.

[42] Young Truong, Xiaodong Lin, and Chris Beecher. Learning a complex metabolomic dataset using random forests and support vector machines. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 835–840. ACM, 2004.

[43] Weibo Wang Jr. Non-uniform language detection in technical writing. 2016.

[44] Jian Xu, Qin Lu, and Zhengzhong Liu. Combining classification with clustering for web person disambiguation. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 637–638. ACM, 2012.

Apéndice A

Detalles de los conjuntos de datos

A.1. Ejemplo de registro en formato XML tomado de la colección Reuters

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="8915" NEWID="4002">
<DATE>11-MAR-1987 18:06:47.22</DATE>
<TOPICS></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>d f BC-FORMER-EMPIRE-OF-CARO 03-11 0107</UNKNOWN>
<TEXT>
    <TITLE>FORMER EMPIRE OF CAROLINA EMP> EXEC SENTENCED</TITLE>
    <DATELINE>     NEW YORK, March 11 - </DATELINE>
    <BODY>
    Mason Benson, former president and
    chief operating officer of Empire of Carolina Inc, a toy maker,
    today was sentenced in Manhattan federal court to a year and
    one day in jail for his involvement in a kickback scheme.
    Benson pleaded guilty to charges of conspiracy, tax
    evasion, filing false corporate tax returns and defrauding the
    company's shareholders. He was also fined 5,000 dlrs.
    </BODY>
</TEXT>
</REUTERS>
```

A.2. Número de registros por clase del conjunto de datos reuters-1

Tabla A.1: Cantidad de registros por clase del conjunto de datos reuters-1 completo, de entrenamiento (75 %) y de prueba (25 %)

Clase	Conjunto de datos completo	Conjunto de datos de entrenamiento (75 %)	Conjunto de datos de prueba (25 %)
acq	244	183	61
alum	5	4	1
bop	5	4	1
carcass	2	2	0
cocoa	12	9	3
coffee	7	6	1
copper	6	5	1
cotton	2	2	0
cpi	16	12	4
cpu	3	3	0
crude	39	30	9
earn	515	387	128
fuel	1	1	0
gas	1	1	0
gnp	21	16	5
gold	16	12	4
grain	7	6	1
heat	1	1	0
housing	4	3	1
interest	18	14	4
ipi	6	5	1
iron-steel	2	2	0
jet	1	1	0
jobs	4	3	1
l-cattle	1	1	0
livestock	1	1	0
lumber	2	2	0
meal-feed	2	2	0
money-fx	21	16	5
money-supply	10	8	2
nat-gas	5	4	1
oilseed	1	1	0
pet-chem	2	2	0
reserves	3	3	0
retail	4	3	1
rubber	6	5	1
ship	11	9	2
silver	2	2	0
sugar	9	7	2
tin	1	1	0
trade	34	26	8
veg-oil	4	3	1
wpi	5	4	1
zinc	1	1	0
Total	1063	813	250

A.3. Número de registros por clase del conjunto de datos reuters-2

Tabla A.2: Cantidad de registros por clase del conjunto de datos reuters-2 completo, de entrenamiento (75 %) y de prueba (25 %)

Clase	Conjunto de datos completo	Conjunto de datos de entrenamiento	Conjunto de datos de prueba
algeria	1	1	0
argentina	5	4	1
aruba	1	1	0
australia	23	18	5
austria	4	3	1
bangladesh	4	3	1
belgium	17	13	4
bolivia	1	1	0
brazil	18	14	4
burma	1	1	0
canada	89	67	22
chile	2	2	0
china	19	15	4
colombia	1	1	0
cuba	1	1	0
denmark	5	4	1
ecuador	10	8	2
egypt	1	1	0
finland	2	2	0
france	34	26	8
ghana	2	2	0
haiti	2	2	0
hong-kong	3	3	0
hungary	5	4	1
india	2	2	0
indonesia	8	6	2
iran	1	1	0
israel	1	1	0
italy	7	6	1
jamaica	2	2	0
japan	49	37	12
jordan	1	1	0
malaysia	3	3	0
morocco	2	2	0
netherlands	21	16	5
new-zealand	6	5	1
nigeria	2	2	0
norway	1	1	0
pakistan	2	2	0
peru	3	3	0
philippines	1	1	0
portugal	3	3	0
saudi-arabia	4	3	1
singapore	3	3	0
south-africa	7	6	1
south-korea	1	1	0
spain	5	4	1
sweden	14	11	3
switzerland	25	19	6
taiwan	5	4	1
thailand	1	1	0
turkey	4	3	1
uganda	1	1	0
uk	137	103	34
usa	1033	775	258
ussr	4	3	1
venezuela	2	2	0
west-germany	28	21	7
zambia	1	1	0
Total	1641	1252	389

Apéndice B

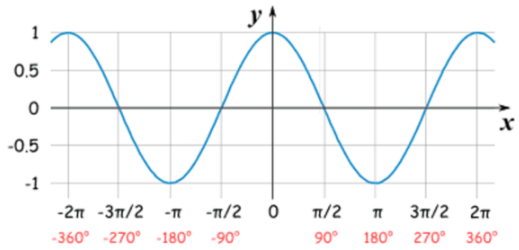
Detalles de los algoritmos

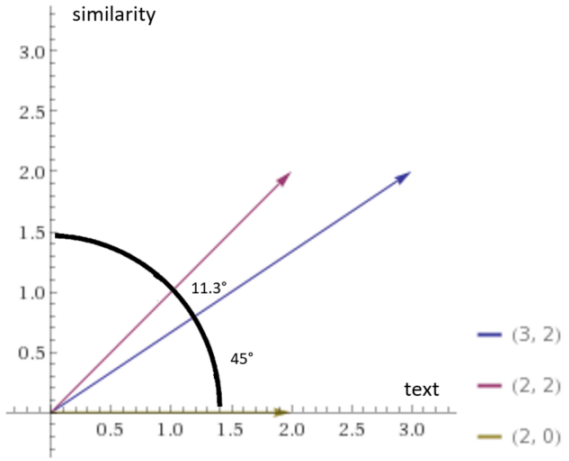
B.1. Ejemplo de similitud Coseno

En la figura B-1 se muestra un ejemplo de representación vectorial para tres documentos y dos palabras, así como el gráfico de la función coseno. La tabla de la izquierda muestra la matriz de palabras tomando en cuenta la frecuencia, mientras que en el gráfico de la derecha se representa cada documento como un vector en R^2. Nótese que, con respecto al vector (3,2), el ángulo formado por el vector (2,2) es menor que el formado por (2,0), lo que significa que el primer y el segundo documento son menos distantes entre sí. En el gráfico inferior izquierdo, se puede observar cómo la función coseno asigna valores entre -1 y 1, de forma que un ángulo cercano a 0 grados obtiene mayor puntaje, mientras que un ángulo cercano a 180 grados obtiene puntaje más bajo. En este ejemplo, la función de similitud coseno asignaría un valor mayor al comparar el primer y el segundo documento, que al comparar el primero con el tercero.

Figura B-1: Ejemplo gráfico de similitud coseno para tres documentos y dos palabras

Document	text	similarity	Vector
text similarity similarity text text	3	2	(3,2)
text similarity text similarity	2	2	(2,2)
text text	2	0	(2,0)





B.2. Definición de la Distribución Dirichlet

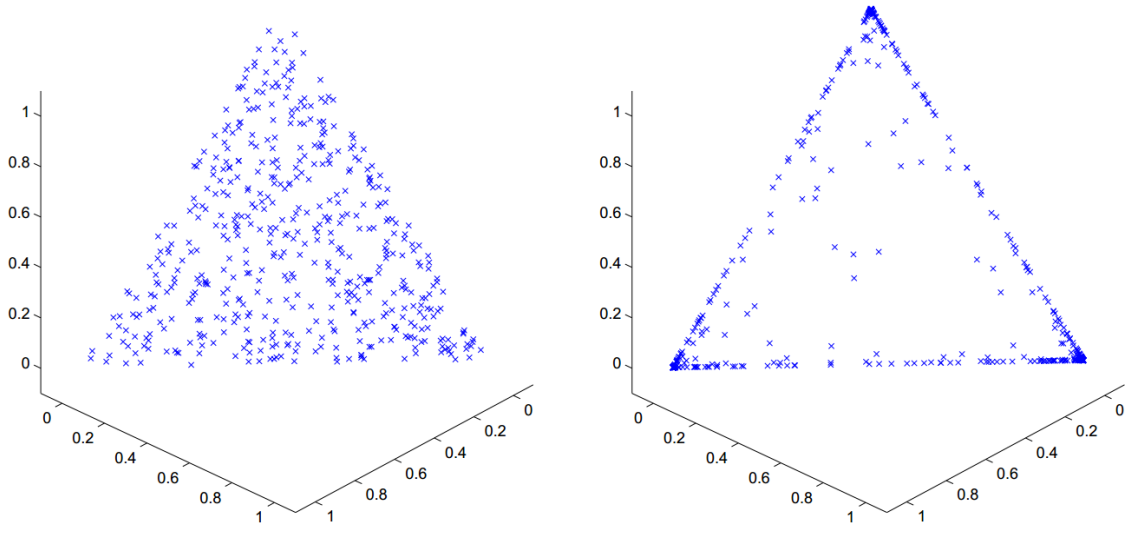
Sea $Q = [Q_1, Q_2, ..., Q_k]$ una función de probabilidad aleatoria, donde $Q \geq 0$ para $i = 1, 2, ..., k$ y $\sum_{i=1}^{k} Q_i = 1$. Adicionalmente, se tiene que $\alpha = [\alpha_1, \alpha_2, ..., \alpha_k]$, con $\alpha > 0$ para todo i, siendo $\alpha_0 = \sum_{i=1}^{k} \alpha_i$. Entonces, se dice que Q tiene una distribucion Dirichlet con parámetro α, denotado por $Q \sim Dir(\alpha)$, si se cumple que $f(q; \alpha) = 0$ cuando q no es una función de probabilidad, y si q sí es una función de probabilidad entonces

$$f(q; \alpha) = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} q_i^{\alpha_i - 1} \tag{B.1}$$

donde $\Gamma(s)$ es la función gama (definida como una generalización de la función factorial). Siendo $m = \alpha/\alpha_0$ la media de la distribución Dirichlet [13].

B.3. Comportamiento para diferentes valores de alfa en la distribución Dirichlet

Figura B-2: Gráfico del comportamiento de los elementos en una distribución Dirichlet para valores de alfa 0.1 y 1 sobre el simplex de probabilidad en R^3



$\alpha = [1, 1, 1]$ $\alpha = [.1, .1, .1]$

Apéndice C

Selección de parámetros

C.1. Paramétros utilizados por los diferentes algoritmos

Tabla C.1: Paramétros seleccionados para ejecutar los diferentes algoritmos

Algoritmo	**Parámetro**	**Valor(es)**
KNN	k	1 y 11
Bagging	tamaño de muestra	0.15
	número de atributos	1.0
LDA	alfa	0.2 | 0.5 | 0.8
	número de temas	44 y 59
	número de iteraciones E	20
	número de iteraciones M	50

C.2. Comportamiento de la exactitud para distintos valores de k

Figura C-1: Gráfico que muestra la tendencia de la precisión para diferentes valores de k en reuters-1.

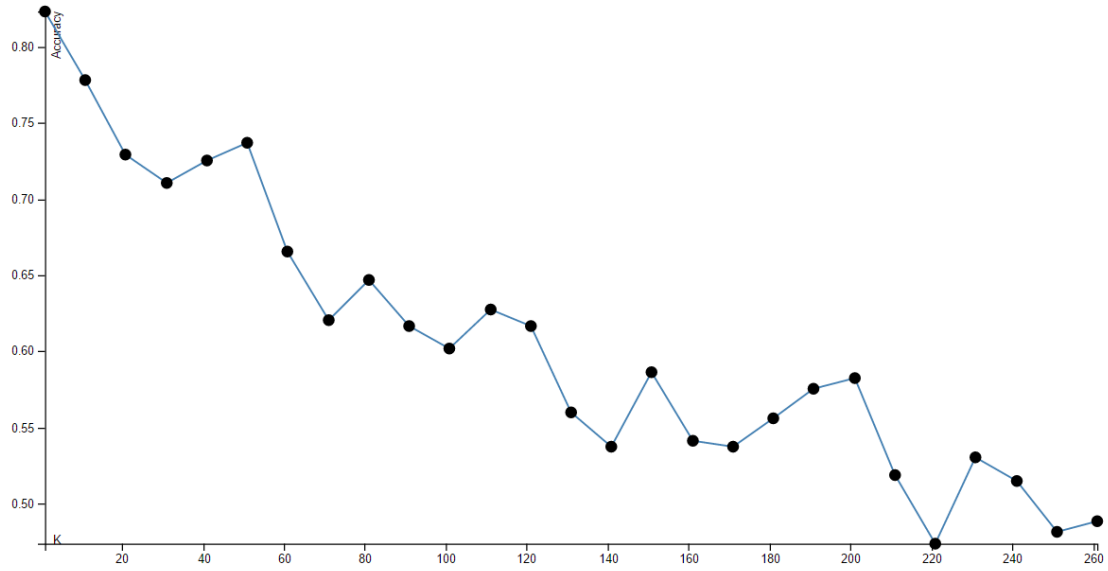
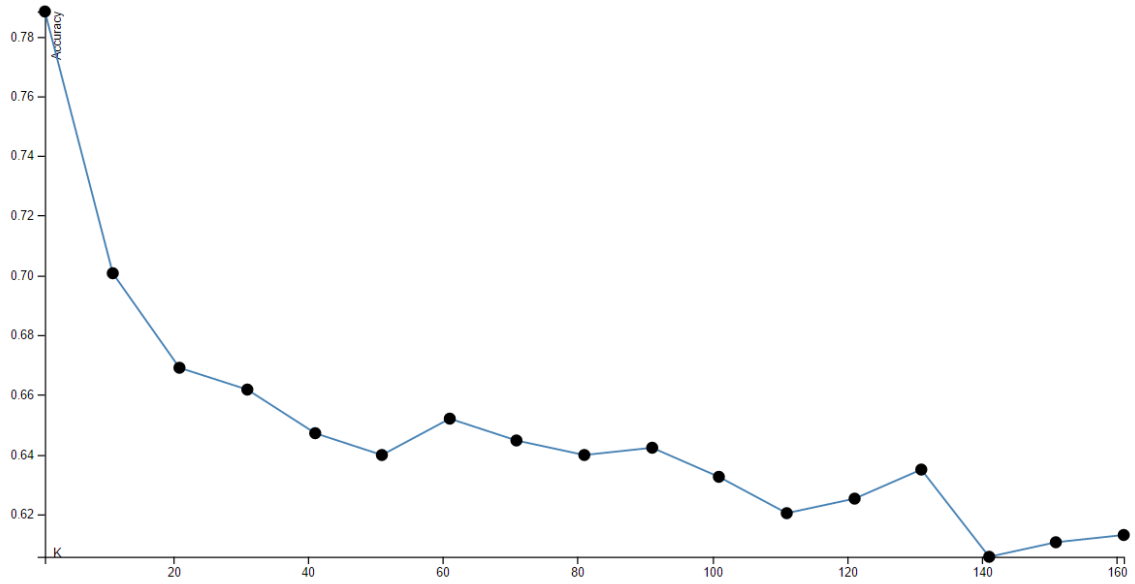


Figura C-2: Gráfico que muestra la tendencia de la precisión para diferentes valores de k en reuters-2.



Apéndice D

Resultados y análisis complementarios

D.1. Resultados de los experimentos con KNN sin metodo agregado

Tabla D.1: Resultados obtenidos para cada experimento utilizando KNN sin *bagging*, por conjunto de datos, valor de k y combinación representación-métrica

Conjunto de datos	k	Representación / Metrica	Exactitud	Precisión	Sensibilidad	F1
reuters-1	1	LDA alfa=0.2 / Coseno	0.8	0.804452	0.8	0.792715
reuters-1	1	LDA alfa=0.5 / Coseno	0.8	0.821869	0.8	0.803953
reuters-1	1	LDA alfa=0.8 / Coseno	0.816	0.815385	0.816	0.811214
reuters-1	1	LDA alfa=0.2 / Divergencia KL	0.8	0.797469	0.8	0.789491
reuters-1	1	LDA alfa=0.5 / Divergencia KL	0.768	0.776959	0.768	0.759982
reuters-1	1	LDA alfa=0.8 / Divergencia KL	0.8	0.797469	0.8	0.789491
reuters-1	1	Vector de palabras con TF-IDF / Coseno	0.808	0.834513	0.808	0.808946
reuters-1	1	Vector de palabras sin TF-IDF / Coseno	0.856	0.860481	0.856	0.852386
reuters-1	1	Vector de palabras sin TF-IDF / Jaccard	**0.9**	**0.900628**	**0.9**	**0.893534**
reuters-1	11	LDA alfa=0.2 / Coseno	0.796	0.779623	0.796	0.780964
reuters-1	11	LDA alfa=0.5 / Coseno	0.788	0.754407	0.788	0.762225
reuters-1	11	LDA alfa=0.8 / Coseno	0.78	0.758448	0.78	0.757489
reuters-1	11	LDA alfa=0.2 / Divergencia KL	0.828	0.796299	0.828	0.808493
reuters-1	11	LDA alfa=0.5 / Divergencia KL	0.824	0.797579	0.824	0.798982
reuters-1	11	LDA alfa=0.8 / Divergencia KL	0.804	0.760077	0.804	0.773005
reuters-1	11	Vector de palabras con TF-IDF / Coseno	0.816	0.803179	0.816	0.797044
reuters-1	11	Vector de palabras sin TF-IDF / Coseno	0.844	0.807863	0.844	0.821831
reuters-1	11	Vector de palabras sin TF-IDF / Jaccard	**0.868**	**0.836532**	**0.868**	**0.845142**
reuters-2	1	LDA alfa=0.2 / Coseno	0.70437	0.719479	0.70437	0.707669
reuters-2	1	LDA alfa=0.5 / Coseno	0.714653	0.750129	0.714653	0.730143
reuters-2	1	LDA alfa=0.8 / Coseno	0.719794	0.748855	0.719794	0.729776
reuters-2	1	LDA alfa=0.2 / Divergencia KL	0.709512	0.741322	0.709512	0.716883
reuters-2	1	LDA alfa=0.5 / Divergencia KL	0.748072	0.765898	0.748072	0.751746
reuters-2	1	LDA alfa=0.8 / Divergencia KL	0.750643	0.779055	0.750643	0.76193
reuters-2	1	Vector de palabras con TF-IDF / Coseno	0.830334	0.847533	0.830334	0.835109
reuters-2	1	Vector de palabras sin TF-IDF / Coseno	0.812339	0.822891	0.812339	0.80835
reuters-2	1	Vector de palabras sin TF-IDF / Jaccard	**0.845758**	**0.851068**	**0.845758**	**0.845541**
reuters-2	11	LDA alfa=0.2 / Coseno	0.74036	0.638793	0.74036	0.682275
reuters-2	11	LDA alfa=0.5 / Coseno	0.701799	0.629419	0.701799	0.662099
reuters-2	11	LDA alfa=0.8 / Coseno	0.706941	0.646742	0.706941	0.664348
reuters-2	11	LDA alfa=0.2 / Divergencia KL	0.745501	0.67593	0.745501	0.700212
reuters-2	11	LDA alfa=0.5 / Divergencia KL	0.74036	0.665861	0.74036	0.68856
reuters-2	11	LDA alfa=0.8 / Divergencia KL	0.755784	0.705765	0.755784	0.716822
reuters-2	11	Vector de palabras con TF-IDF / Coseno	**0.820051**	**0.791033**	**0.820051**	**0.787891**
reuters-2	11	Vector de palabras sin TF-IDF / Coseno	0.760925	0.688958	0.760925	0.713346
reuters-2	11	Vector de palabras sin TF-IDF / Jaccard	0.807198	0.775127	0.807198	0.77375

D.2. Resultados de los experimentos utilizando un conjunto de datos uniformemente etiquetado

Figura D-1: Gráfico. Valores de exactitud obtenidos al ejecutar los experimentos utilizando el conjunto de datos reuters-1 modificado con k=1

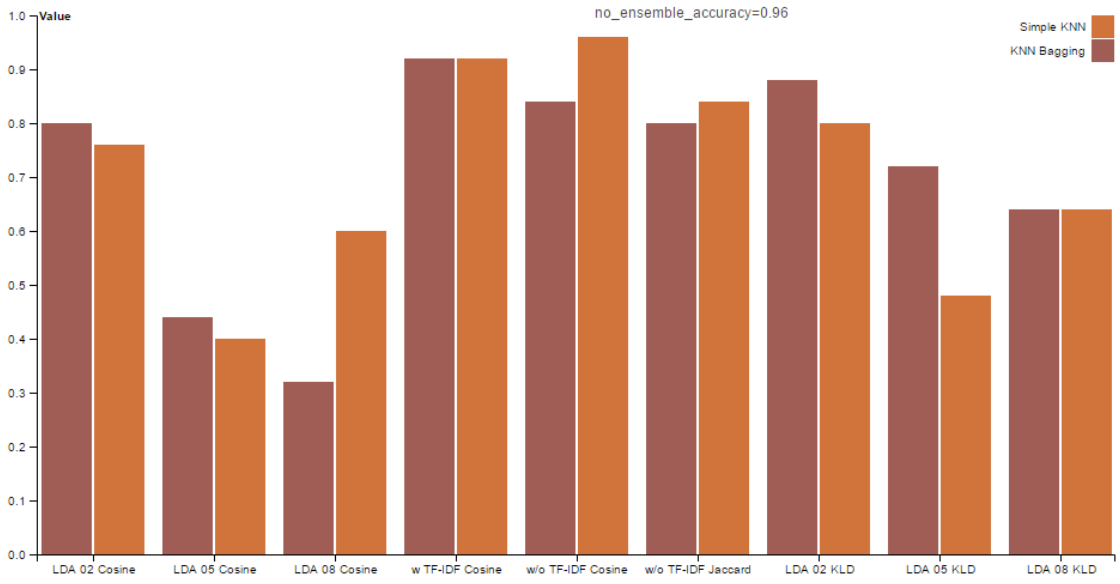
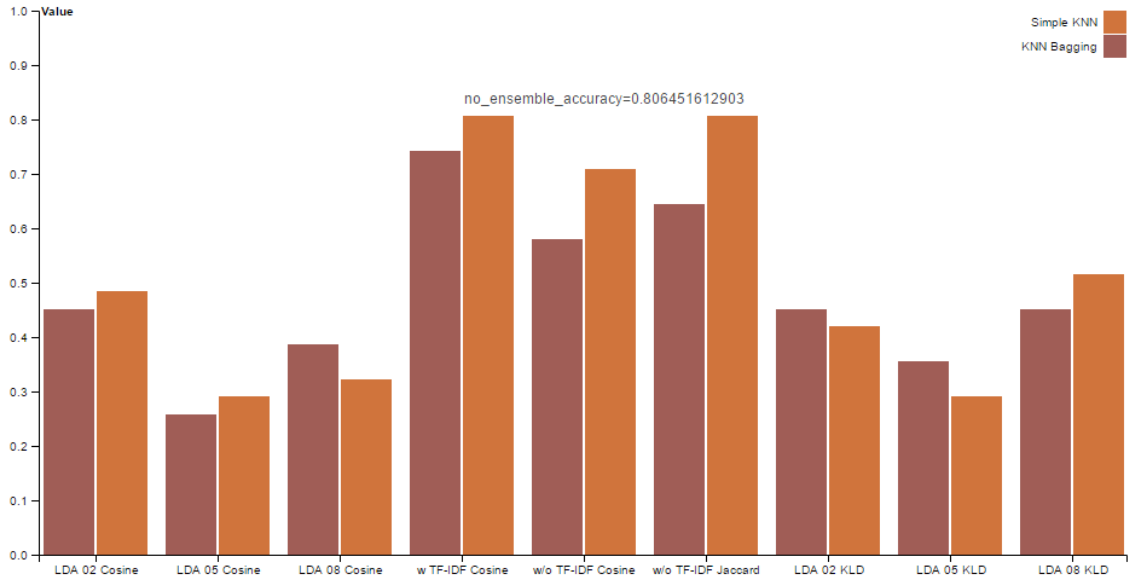


Figura D-2: Gráfico. Valores de exactitud obtenidos al ejecutar los experimentos utilizando el conjunto de datos reuters-2 modificado con k=1



D.3. Imputaciones obtenidas al ejecutar el experimento de LDA y Divergencia KL para k=1

Clase Esperada	Clase Imputada	Tamaño del texto (bytes)	Clase Esperada	Clase Imputada	Tamaño del texto
sugar	sugar	6285	earn	earn	382
crude	crude	3789	earn	earn	378
ship	ship	3612	earn	earn	369
earn	earn	3401	earn	earn	368
acq	acq	3178	earn	earn	363
acq	acq	2822	acq	acq	355
interest	interest	2783	earn	earn	355
earn	earn	2537	earn	earn	353
crude	crude	2526	earn	earn	353
trade	trade	2163	acq	acq	351
crude	crude	2161	earn	earn	348
crude	crude	1880	earn	earn	348
crude	crude	1791	acq	acq	342
gnp	gnp	1786	earn	earn	341
acq	acq	1752	earn	earn	337
money-fx	money-fx	1736	earn	earn	334
trade	trade	1633	earn	earn	332
crude	crude	1589	earn	earn	330
trade	trade	1529	earn	earn	330
acq	acq	1509	earn	earn	328
crude	crude	1506	acq	acq	327
acq	acq	1500	earn	earn	316
cocoa	cocoa	1473	earn	earn	316
acq	acq	1367	earn	earn	311
acq	acq	1291	earn	earn	299

earn	earn	1291	earn	earn	298
acq	acq	1166	acq	acq	290
trade	trade	1129	earn	earn	289
gold	gold	1093	acq	acq	289
earn	earn	1032	earn	earn	287
acq	acq	995	acq	acq	284
earn	earn	953	earn	earn	283
acq	acq	939	earn	earn	280
earn	earn	921	earn	earn	279
earn	earn	876	earn	earn	276
gold	gold	839	earn	earn	273
acq	acq	834	earn	earn	272
earn	earn	829	earn	earn	271
gold	gold	824	earn	earn	270
earn	earn	816	earn	earn	268
acq	acq	768	earn	earn	267
acq	acq	759	earn	earn	266
acq	acq	726	earn	earn	262
earn	earn	709	earn	earn	261
acq	acq	681	earn	earn	258
earn	earn	675	acq	acq	257
earn	earn	670	acq	acq	249
gnp	gnp	666	earn	earn	246
acq	acq	665	earn	earn	244
earn	earn	636	earn	earn	239
cocoa	cocoa	631	earn	earn	238
earn	earn	626	earn	earn	231
earn	earn	625	earn	earn	229
earn	earn	624	earn	earn	225
rubber	rubber	624	earn	earn	222
acq	acq	621	earn	earn	222
acq	acq	617	earn	earn	221
earn	earn	616	earn	earn	212
earn	earn	593	earn	earn	211
earn	earn	591	earn	earn	204
earn	earn	590	earn	earn	204
acq	acq	587	money-fx	money-fx	203
acq	acq	577	earn	earn	199
earn	earn	569	earn	earn	190
earn	earn	563	earn	earn	188
coffee	coffee	562	earn	earn	183
earn	earn	561	earn	earn	170
earn	earn	561	earn	earn	163
acq	acq	558	earn	earn	161
acq	acq	549	earn	earn	161
acq	acq	547	earn	earn	160
earn	earn	543	earn	earn	159
money-fx	money-fx	530	earn	earn	155
earn	earn	522	earn	earn	147
earn	earn	511	earn	earn	146
acq	acq	499	earn	earn	144
acq	acq	495	earn	earn	143
acq	acq	485	earn	earn	142
earn	earn	484	earn	earn	140
acq	acq	481	earn	earn	139
acq	acq	478	earn	earn	137
acq	acq	469	earn	earn	134
acq	acq	455	earn	earn	131

acq	acq	452	earn	earn	125
acq	acq	449	earn	earn	124
earn	earn	449	earn	earn	115
acq	acq	448	cpi	cpi	101
acq	acq	441	acq	acq	91
earn	earn	435	acq	acq	90
acq	acq	426	crude	crude	89
earn	earn	418	earn	earn	87
earn	earn	415	acq	acq	85
earn	earn	412	earn	earn	85
earn	earn	410	acq	acq	83
earn	earn	409	acq	acq	83
earn	earn	404	earn	earn	83
earn	earn	402	earn	earn	79
earn	earn	399	earn	earn	71
earn	earn	398	earn	earn	69
earn	earn	394	earn	earn	53
earn	earn	385			

Tabla D.2: Imputaciones correctas obtenidas al ejecutar el experimento de LDA y Divergencia KL para KNN simple con k=1

Clase Esperada	Clase Imputada	Tamaño del texto (bytes)	Clase Esperada	Clase Imputada	Tamaño del texto
earn	acq	4352	earn	acq	635
gnp	earn	2890	ipi	money-supply	623
interest	earn	2634	acq	money-fx	597
earn	crude	2166	acq	interest	587
money-fx	jobs	1943	trade	cpi	582
copper	gold	1940	acq	earn	579
acq	earn	1853	retail	trade	543
trade	lumber	1791	acq	interest	506
bop	earn	1786	jobs	ipi	493
interest	earn	1721	earn	gnp	482
nat-gas	gas	1710	acq	earn	473
cocoa	rubber	1280	wpi	trade	460
trade	sugar	1220	acq	earn	423
money-fx	trade	1184	acq	earn	422
money-supply	reserves	1166	veg-oil	oilseed	391
cpi	wpi	1105	cpi	trade	386
housing	cpu	1028	acq	earn	368
alum	acq	1020	acq	earn	191
cpi	coffee	878	money-supply	reserves	106
acq	earn	754	crude	acq	102
grain	trade	737	trade	cpi	101
gold	acq	723	sugar	acq	96
interest	crude	693	gnp	interest	87
acq	earn	669	gnp	cpi	82
ship	zinc	647			

Tabla D.3: Imputaciones incorrectas obtenidas al ejecutar el experimento de LDA y Divergencia KL para KNN simple con k=1

D.4. Resultados de los experimentos sobre documentos de textos cortos

Tabla D.4: Resultados obtenidos utilizando k=1 y documentos cortos (menores a 150 bytes) tomados de reuters-1

Algoritmo	Representación / Metrica	Exactitud	Precisión	Sensibilidad	F1
Bagging	LDA alfa=0.2 / Coseno	0.727273	0.666667	0.727273	0.6892
Bagging	LDA alfa=0.5 / Coseno	0.666667	0.600449	0.666667	0.602376
Bagging	LDA alfa=0.8 / Coseno	0.787879	0.783261	0.787879	0.76985
Bagging	LDA alfa=0.2 / Divergencia KL	0.787879	0.66929	0.787879	0.718615
Bagging	LDA alfa=0.5 / Divergencia KL	0.757576	0.656566	0.757576	0.697643
Bagging	LDA alfa=0.8 / Divergencia KL	**0.818182**	**0.73913**	**0.818182**	**0.775899**
Bagging	Vector de palabras con TF-IDF / Coseno	0.636364	0.439394	0.636364	0.506605
Bagging	Vector de palabras sin TF-IDF / Coseno	0.757576	0.693122	0.757576	0.710344
Bagging	Vector de palabras sin TF-IDF / Jaccard	0.69697	0.587413	0.69697	0.620444
KNN simple	LDA alfa=0.2 / Coseno	0.727273	0.651974	0.727273	0.683261
KNN simple	LDA alfa=0.5 / Coseno	0.787879	0.733766	0.787879	0.759323
KNN simple	LDA alfa=0.8 / Coseno	**0.818182**	**0.799423**	**0.818182**	**0.804747**
KNN simple	LDA alfa=0.2 / Divergencia KL	0.727273	0.651974	0.727273	0.683261
KNN simple	LDA alfa=0.5 / Divergencia KL	0.787879	0.762626	0.787879	0.774105
KNN simple	LDA alfa=0.8 / Divergencia KL	0.757576	0.708827	0.757576	0.72876
KNN simple	Vector de palabras con TF-IDF / Coseno	0.69697	0.594406	0.69697	0.624757
KNN simple	Vector de palabras sin TF-IDF / Coseno	0.787879	0.772727	0.787879	0.769054
KNN simple	Vector de palabras sin TF-IDF / Jaccard	0.787879	0.772727	0.787879	0.777778

D.5. Resultados de los experimentos al incrementar el número de temas en LDA

Tabla D.5: Resultados obtenidos utilizando k=1 y diferentes números de temas al generar las representacines con LDA

Representación / Metrica	Variable	Algoritmo	44 temas	100 temas	200 temas
LDA alfa=0.2 / Coseno	**Exactitud**	**bagging**	0.792	0.82	0.808
LDA alfa=0.8 / Coseno			0.776	0.776	-
LDA alfa=0.2 / Divergencia KL			0.804	0.832	0.848
LDA alfa=0.8 / Divergencia KL			0.804	0.724	-
LDA alfa=0.2 / Coseno		**KNN simple**	0.8	0.792	0.804
LDA alfa=0.8 / Coseno			0.816	0.812	-
LDA alfa=0.2 / Divergencia KL			0.8	0.804	0.832
LDA alfa=0.8 / Divergencia KL			0.8	0.804	-
LDA alfa=0.2 / Coseno	**F1**	**bagging**	0.777647	0.804226	0.783654
LDA alfa=0.8 / Coseno			0.766546	0.7511	-
LDA alfa=0.2 / Divergencia KL			0.772981	0.80398	0.820342
LDA alfa=0.8 / Divergencia KL			0.772981	0.696903	-
LDA alfa=0.2 / Coseno		**KNN simple**	0.792715	0.802081	0.797725
LDA alfa=0.8 / Coseno			0.811214	0.819934	-
LDA alfa=0.2 / Divergencia KL			0.789491	0.80198	0.829696
LDA alfa=0.8 / Divergencia KL			0.789491	0.790334	-

Apéndice E

Pruebas estadísticas adicionales

E.1. Análisis del puntaje F1 con respecto a la combinación representación-métrica utilizando *bagging*

Resultado de prueba de normalidad Lillifors sobre valores de puntaje F1 para $k = 1$ y $k = 11$:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cF1
D = 0.094876, p-value = 0.5674
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de puntaje F1:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
representation_metric	8	0.02341	0.002926	0.526	0.826

Prueba de ANOVA muestra que no hay diferencias significativas entre las poblaciones ($p > 0.05$).

E.2. Análisis para diferentes valores de k

Resultado de prueba de normalidad Lillifors sobre valores de exactitud para todos los valores de k:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cAccuracy
D = 0.083755, p-value = 0.01508
```

Valores de exactitud no son normales ($p < 0.05$).

Resultado de prueba Kruskal-Wallis sobre valores de exactitud con respecto a diferentes valores de k:

```
Kruskal-Wallis chi-squared = 23.18, df = 3,
p-value = 3.704e-05
```

Prueba de Kruskal-Wallis muestra diferencias significativas entre las poblaciones ($p < 0.05$).

Resultado de prueba post-hoc de pares Nemenyi con respecto a diferentes valores de k:

```
   1        11       21
11 0.3498   -        -
21 0.0071   0.3995   -
31 3.2e-05  0.0198   0.5424
```

Valor de $k = 1$ muestra diferencias estadísticas significativas en la exactitud con respecto a $k = 21$ y $k = 31$.

E.3. Análisis para bagging vs KNN sin método agregado

Resultado de prueba de normalidad Lillifors sobre valores de exactitud para $k = 1$ y $k = 11$:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cAccuracy
D = 0.093399, p-value = 0.1246
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de exactitud con respecto a bagging y KNN sin método agregado:

```
             Df  Sum Sq  Mean Sq F value  Pr(>F)
algorithm     1 0.02408 0.024079   10.38 0.00194
```

Prueba de ANOVA muestra que sí hay diferencias significativas entre las poblaciones ($p < 0.05$). Esto significa que, en cuanto a exactitud, se obtuvieron mejores resultados sin utilizar bagging.

Resultado de prueba de normalidad Lillifors sobre valores de F1 ponderado para $k = 1$ y $k = 11$:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cF1
D = 0.10086, p-value = 0.06685
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de F1 ponderado con respecto a bagging y KNN sin método agregado:

```
             Df  Sum Sq Mean Sq F value  Pr(>F)
```

```
algorithm     1 0.09854 0.09854      23.5 7.27e-06
```

Prueba de ANOVA muestra que sí hay diferencias significativas entre las poblaciones ($p < 0.05$). Esto significa que, en cuanto a puntaje F1, se obtuvieron mejores resultados sin utilizar bagging.

E.4. Análisis para diferentes valores de alfa en LDA

Resultado de prueba de normalidad Lillifors sobre valores de exactitud obtenidos en experimentos que involucraron LDA:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cAccuracy
D = 0.11696, p-value = 0.09821
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de exactitud con respecto a diferentes valores del parámetro alfa:

```
             Df  Sum Sq   Mean Sq F value Pr(>F)
fact          2 0.00037 0.0001867   0.104  0.901
```

Prueba de ANOVA muestra que no hay diferencias significativas entre las poblaciones ($p > 0.05$).

Resultado de prueba de normalidad Lillifors sobre valores de F1 ponderado con respecto a diferentes valores del parámetro alfa:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cF1
D = 0.10911, p-value = 0.1631
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de F1 ponderado con respecto a diferentes valores del parámetro alfa:

```
            Df  Sum Sq  Mean Sq F value Pr(>F)
alpha        1 0.00033 0.000332   0.086   0.77
```

Prueba de ANOVA muestra que tampoco hay diferencias significativas entre las poblaciones basadas en F1 ponderado ($p > 0.05$).

E.5. Análisis basado en representación-métrica de los datos para cualquier algoritmo

Resultado de prueba de normalidad Lillifors sobre valores de exactitud obtenidos en experimentos utilizando $k <= 11$, bagging y sin método agregado, alfa=0.8:

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  cAccuracy
D = 0.093399, p-value = 0.1246
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de exactitud:

```
                       Df  Sum Sq  Mean Sq F value Pr(>F)
representation_metric   8 0.04619 0.005774   2.592 0.0163
```

Prueba de ANOVA muestra diferencias significativas entre las poblaciones ($p < 0.05$).

Resultado de prueba post-hoc de pares TukeyHSD sobre valores de exactitud:

```
                                                  p adj
LDA Cosine 5-LDA Cosine 2                     1.0000000
LDA Cosine 8-LDA Cosine 2                     1.0000000
LDA KLD 2-LDA Cosine 2                        0.9999894
```

LDA KLD 5–LDA Cosine 2	0.9999996
LDA KLD 8–LDA Cosine 2	1.0000000
Words Vector w TF–IDF Cosine 0–LDA Cosine 2	0.3466361
Words Vector w/o TF–IDF Cosine 0–LDA Cosine 2	0.5388277
Words Vector w/o TF–IDF Jaccard 0–LDA Cosine 2	0.2070753
LDA Cosine 8–LDA Cosine 5	1.0000000
LDA KLD 2–LDA Cosine 5	0.9998885
LDA KLD 5–LDA Cosine 5	0.9999876
LDA KLD 8–LDA Cosine 5	0.9999983
Words Vector w TF–IDF Cosine 0–LDA Cosine 5	0.2757601
Words Vector w/o TF–IDF Cosine 0–LDA Cosine 5	0.4520056
Words Vector w/o TF–IDF Jaccard 0–LDA Cosine 5	0.1575729
LDA KLD 2–LDA Cosine 8	0.9998298
LDA KLD 5–LDA Cosine 8	0.9999780
LDA KLD 8–LDA Cosine 8	0.9999965
Words Vector w TF–IDF Cosine 0–LDA Cosine 8	0.2615632
Words Vector w/o TF–IDF Cosine 0–LDA Cosine 8	0.4336155
Words Vector w/o TF–IDF Jaccard 0–LDA Cosine 8	0.1480626
LDA KLD 5–LDA KLD 2	1.0000000
LDA KLD 8–LDA KLD 2	0.9999999
Words Vector w TF–IDF Cosine 0–LDA KLD 2	0.5806013
Words Vector w/o TF–IDF Cosine 0–LDA KLD 2	0.7738816
Words Vector w/o TF–IDF Jaccard 0–LDA KLD 2	0.3968523
LDA KLD 8–LDA KLD 5	1.0000000
Words Vector w TF–IDF Cosine 0–LDA KLD 5	0.4975588
Words Vector w/o TF–IDF Cosine 0–LDA KLD 5	0.6989219
Words Vector w/o TF–IDF Jaccard 0–LDA KLD 5	0.3243980
Words Vector w TF–IDF Cosine 0–LDA KLD 8	0.4423243
Words Vector w/o TF–IDF Cosine 0–LDA KLD 8	0.6440095
Words Vector w/o TF–IDF Jaccard 0–LDA KLD 8	0.2794663
Words Vector w/o TF–IDF Cosine 0–Words Vector w TF–IDF Cosine 0	0.9999975
Words Vector w/o TF–IDF Jaccard 0–Words Vector w TF–IDF Cosine 0	0.9999986
Words Vector w/o TF–IDF Jaccard 0–Words Vector w/o TF–IDF Cosine 0	0.9996249

Resultado de prueba de normalidad Lillifors sobre valores de F1 ponderados obtenidos en experimentos utilizando $k <= 11$, bagging y sin método agregado, alfa=0.8:

```
Lilliefors (Kolmogorov–Smirnov) normality test
data:  cF1
D = 0.10086, p–value = 0.06685
```

Valores de exactitud son normales ($p > 0.05$).

Resultado de prueba ANOVA sobre valores de exactitud:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
representation_metric	8	0.0517	0.006463	1.196	0.316

Prueba de ANOVA muestra que no hay diferencias significativas entre las poblaciones ($p > 0.05$).

Estas pruebas estadísticas tampoco aportan evidencia significativa para determinar si una combinación fue mejor o peor que las demás. La prueba de anova para la exactitud sugiere diferencias, sin embargo, en la prueba de pares TukeyHSD no se muestran valores p significativos para apoyar esto (La mayor diferencia se observa en los resultados obtenidos usando vector de palabras sin TF-IDF y Jaccard en comparación con los obtenidos utilizando LDA y Coseno).

Apéndice F

Código fuente

F.1. Implementación en Python del programa principal

A continuación se muestra el código en Python del algoritmo principal implementado para ejecutar los experimentos.

Para su ejecucion, el programa requiere la existencia de los conjuntos de documentos preprocesados. Adicionalmente, los parámetros deben definirse en un archivo Parameters.py, tal y como se especifica en la documentación del *script*. Dependiendo del origen de los documentos y del formato de salida deseado, el archivo de parámetros debe contener la implementación de las funciones para leer documentos y escribir resultados.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import DistanceMetric
from sklearn.metrics.pairwise import cosine_similarity
from scipy.stats import entropy

#############################################################################
# Parameters is a custom file containing the required parameters to run the program:
# datasetRepresentations: List of objects, each containing the name of the train
#                         and the test files, and the representation type
# metricNames: List of metrics to be used in the experiments. Supported values:
#              Jaccard, Cosine, Divergence-KL
# techniques: Description of the ensemble algorithm(s) to be used. Supported values:
#             No ensemble, bagging
# k: Initial value of k to be used in the KNN algorithm
# increment: Value to be added to k on each iteration, giving a new k for the KNN
#            algorithm
# maxK: Highest k to be used in the experiment
# maxSamples: Sample size to be used by the bagging algorithm
# readFile: Function to read a file given the filename, returning a list of parsed
#           documents (id, text, words/topics vector(comma separated) and class)
# writeResult: Funtion to write the result of the experiment given the required
#              values
#############################################################################
from Parameters import datasetRepresentations, metricNames, techniques, k, \
    increment, maxK, maxSamples, readFile, writeResult

# Jaccard distance
def customJaccard(x,y):
    Xvalue = [x,y]
    dist = DistanceMetric.get_metric('jaccard')
    jaccardResult = dist.pairwise(Xvalue)
    return jaccardResult[0][1]

# Cosine distance
def customCosine(x,y):
    cosResult = cosine_similarity(x,y)
    return (1-(cosResult[0] + 1)/2)

# KL Divergence distance
def customKLD(x,y):
    newX = [0.000001 if e == 0 else e for e in x]
    newY = [0.000001 if e == 0 else e for e in y]
    KLResult = entropy(newX,newY)
    return KLResult
```

```
original_k = k

#Iterates over dataset representations
for (datasetRepresentation) in datasetRepresentations:
    # name of the file containing the train documents
    datasetRepresentationTrain = datasetRepresentation[0]
    # name of the file containing the test documents
    datasetRepresentationTest = datasetRepresentation[1]
    # 'with TF-IDF', 'without TF-IDF' or 'LDA'
    representationType = datasetRepresentation[2]

    #Iterates over distance metrics
    for (metricName) in metricNames:

        # Only certain combinations are allowed
        if (metricName == "Jaccard" and representationType == "without_TF-IDF") \
                or (metricName == "Divergence-KL" and representationType == "LDA") \
                or metricName == "Cosine":
            documentsTrain = readFile(datasetRepresentationTrain)
            documentsTest = readFile(datasetRepresentationTest)
            k = original_k

            # Runs for different k values
            while (k < maxK):

                # Iterates over techniques
                for (technique) in techniques:
                    vectorsTrain = []
                    vectorsTest = []
                    classesTrain = []
                    classesTest = []
                    documentIdsTrain = []
                    documentIdsTest = []
                    for (document) in documentsTrain:
                        documentIdsTrain.append(document[0])
                        vectorsTrain.append(document[3].split(','))
                        classesTrain.append(document[4])
                    for (document) in documentsTest:
                        documentIdsTest.append(document[0])
                        vectorsTest.append(document[3].split(','))
                        classesTest.append(document[4])

                    funcDist = None
                    if metricName == 'Cosine':
```

```
                funcDist = customCosine
            if metricName == 'Jaccard':
                funcDist = customJaccard
            if metricName == 'Divergence-KL':
                funcDist = customKLD

            #Creates the instance of the KNN classifier
            neigh = KNeighborsClassifier(n_neighbors=k, metric=funcDist)
            # Simple KNN. Only fits model and predicts
            if (technique == "No_ensemble"):
                neigh.fit(vectorsTrain, classesTrain)
                predictions = []
                for (vector) in vectorsTest:
                    predictions.append(neigh.predict(vector))
            # Bagging. Creates Bagging classifier, fits model and predicts
            if (technique == "bagging"):
                bagging = BaggingClassifier(neigh, max_samples=maxSamples,
                                            max_features=1.0)
                bagging.fit(vectorsTrain, classesTrain)
                predictions = []
                for (vector) in vectorsTest:
                    predictions.append(bagging.predict(vector))

            # Final result is written for the experiment
            writeResult(predictions, datasetRepresentation, k, metricName,
                        technique)

        k = k + increment
```

Apéndice G

Recursos en línea

- Sitio web con acceso a la base de datos utilizada en esta investigación. Permite consultar y descargar los conjuntos de datos preprocesados utilizando las distintas técnicas, consultar y descargar los resultados obtenidos en cada experimento, así como interactuar con gráficos de resultados: http://jocan3.com/textsimilarity_dev
- Scripts de R para preprocesamiento de datos mediante vectores de palabras: https://github.com/jocan3/TextDataPreProcessingR
- Programas en Python que implementan el algoritmo KNN, bagging y las métricas de distancia de texto: https://github.com/jocan3/TextKNNPy