ISSN 1870-4069

Derecho comparado asistido por computadora: Procesamiento de lenguaje natural legal

Héctor Alejandro Vargas-Gutiérrez[1], Gerardo Rodriguez-Hernandez[2]

[1] Universidad de Guadalajara,
Centro Universitario de Ciencias Económico Administrativas,
México

[2] Tecnológico de Monterrey,
Escuela de Ingeniería y Ciencias,
México

hecvagu@hotmail.com, gerardo.rodriguez@tec.mx

Resumen. El sistema legal Mexicano cuenta con un amplio conjunto de leyes y regulaciones. Sin embargo, el crecimiento constante del marco normativo ha generado dificultades en el análisis y control de las normas debido a la redacción diversa realizada por diferentes autoridades legislativas, dando lugar a la existencia de legislaciones semánticamente similares pero lexicamente distintas. Este artículo propone un método basado en procesamiento de lenguaje natural o PLN (*Natural Language Processing* NLP por sus siglas en ingles) para ayudar al sistema jurídico mexicano, en particular al derecho comparado, mediante la identificación de expresiones con significados semánticamente similares en las constituciones de los 32 estados mexicanos. Se utilizaron cuatro diferentes transformaciones de texto en espacios vectoriales (*embedding*) (count vectorizer, tf idf, word2vec, BERT) y tres enfoques de agrupamiento (*clustering*) (k medias, GMM, agrupamiento aglomerativo) para el procesamiento de los artículos constitucionales. Se evaluaron diferentes casos de prueba y se observó que la representación en el espacio vectorial Word2Vec ofreció los mejores resultados en general, independientemente del método de agrupamiento utilizado. Sin embargo la combinación puntual que obtuvo el mejor resultado fue BERT con k-medias.

Palabras clave: PLN, procesamiento lenguaje natural legal, agrupamiento, derecho comparado.

Computer-Assisted Comparative LAW: Legal Natural Language Processing

Abstract. The Mexican legal system has a broad set of laws and regulations. However, the growth of the regulatory framework in recent years has generated difficulties in the analysis and control of norms due to the diverse wording used by different authorities, which can generate confusion and legal conflicts. This article proposes a procedure based on natural language processing (NLP) to assist the Mexican legal system, particularly comparative law, by identifying expressions

with semantically similar meanings in the constitutions of the 32 Mexican states. Text transformations in vector spaces and clustering approaches were suggested for the processing of constitutional articles. Laws were collected and preprocessed, and representation techniques in vector spaces, such as Word2Vec, were applied for distance-based clustering. Different test cases were evaluated, and it was observed that the Word2Vec embedding offered the best overall results, regardless of the clustering method used. The best-performing combination was BERT with k-means to 22 groups.

Keywords: NLP, legal natural lenguage processing, clustering, comparative law.

1. Introducción

El derecho regula múltiples sectores de la sociedad, lo cual implica el análisis y creación de grandes cantidades de texto, así como la interpretación del lenguaje legal. En el caso del sistema legal mexicano, mantenerse actualizado en la interpretación de textos complejos supone un gran esfuerzo debido a las numerosas leyes vigentes. El procesamiento de lenguaje natural (PLN) se aplica en diferentes contextos legales para mejorar la eficiencia y automatizar tareas rutinarias [10, 11, 6].

El Derecho Comparado, se enfoca en el estudio de los sistemas legales de distintos ordenamientos jurídicos, de manera comparativa, a fin de identificar las mejores prácticas y soluciones, y mejorar el propio sistema legal. Los abogados y juristas pueden aplicar estos conocimientos para mejorar la calidad y eficacia de la justicia [5].

El sistema legal de México se caracteriza por una amplia colección de leyes y regulaciones, lo que dificulta su análisis y control, generando posibles conflictos legales [4, 9]. Proponemos un modelo que tiene el potencial de ayudar en la labor del derecho comparado a identificar los artículos entre las constituciones de todos los estados más la Constitución Política de los Estados Unidos Mexicanos (CPEUM), que se refieren a un mismo tema, aún cuando sean escritos con diferentes términos.

Este problema de similitud de textos se puede resolver como un problema geométrico de distancias en espacios vectoriales como lo hizo Shahmirzadi, Lugowski y Younge, en su trabajo: Modelos de similitud de textos en espacios vectoriales: un estudio comparativo, donde miden la similitud de patentes con variaciones del modelo de representación en espacios vectorialesTF IDF [12].

Además otros autores como Arnarsson, Frost, Gustavsson, Jirstrand y Malmqvist proponen en su trabajo: Métodos de PLN para la gestión del conocimiento: aplicación agrupamiento de documentos para una búsqueda y agrupación rápidas de documentos de ingeniería, otras transformaciones de texto a espacios vectoriales como doc2vec y para agrupación el modelo asignación latente de Dirichlet (*Latent Dirichlet allocation* LDA) para identificar documentos relacionados en una base de datos de solicitudes de cambios de ingeniería, donde con ayuda de un experto corporativo examinan si los informes pertenecen o no a los grupos generados [8].

En este trabajo se comparan modelos con cuatro métodos de representación en espacios vectoriales y tres métodos de agrupamiento, los cuales son evaluados con

Fig. 1. Distribución de la cantidad de palabras por artículo en el corpus antes y después de limitar la cantidad máxima a 500 palabras.

una métrica que integra cuatro casos de prueba que son proporcionados por un experto en el área legal y otros obtenidos de un tesauro de la suprema corte de justicia de la nación (SCJN).

El objetivo principal de este estudio es explorar los métodos de PLN que mejor describan la similaridad semántica de los artículos de la ley mexicana con respecto a los casos de prueba y compararlos a través de una métrica especializada sencilla para descubrir el modelo que mejor desempeño tiene en esta tarea.

2. Metodología

Como una posible solución al problema del derecho comparado a través del PLN se propone la agrupación de los artículos por sus significado semántico, para lograrlo se transforma el problema de trabajar con textos y palabras a trabajar en espacios vectoriales, agrupando estos artículos de las constituciones con base en su cercanía semántica a través de estas representaciones en espacios vectoriales y su cercanía geométrica, esto quiere decir que dentro de estos grupos se pueden encontrar los artículos que se refieren a temas similares, a pesar de que estos estén expresados con diferentes palabras. El proceso detallado de la solución se describe a continuación:

2.1. Elaboración del corpus de datos

Para el presente estudio de PLN aplicado al problema legal planteado, se recopilaron las 32 constituciones políticas de los estados de México además de la CPEUM, formando un corpus con los artículos que integran estas 33 constituciones las cuales fueron descargadas de las páginas oficiales de los congresos de cada estado incluyendo la CPEUM que fue descargada desde la página oficial de la SCJN.

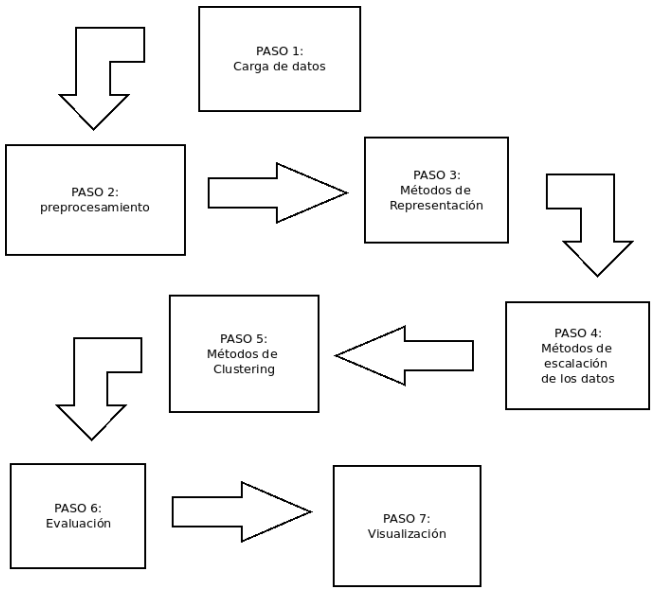

Fig. 2. Clases del modelo de implementación.

En la figura 1 se muestra la longitud de cada artículo medida por el número de palabras contenida en cada uno; La gráfica derecha muestra una amplia variación en el tamaño de los artículos originales, llegando algunos a exceder el tamaño 2300 palabras. En el corpus original el recuento de artículos es de 5051 mientras que el corpus de trabajo con el limite máximo de 500 palabras representado en la gráfica izquierda de la figura 1 es de 14047.

El umbral de 500 palabras se definió como longitud máxima, debido a que es el límite de entrada que acepta uno de los modelos de representación utilizados en este estudio. Por este motivo, se realizó un primer pre-procesamiento o curado de los documentos utilizando una combinación de expresiones regulares y separación semi automática, debido a que no existe un estándar o una norma que regule la forma de construir los artículos entre las constituciones de los estados de la república mexicana.

La separación de estos artículos se hizo tomando en cuenta la jerarquía en las numeraciones de cada constitución, en la mayor parte de estas constituciones la jerarquía más alta la tienen los apartados usualmente en letras del alfabeto romano en mayúsculas, en seguida los listados con números romanos, y al final los listados con letras del alfabeto romano en minúsculas o en números arábigos. Todos estos con variantes de punto, punto y guión, punto y coma, paréntesis, espacio o nada.

2.2. El sistema

Se eligió Python como lenguaje de programación para los experimentos realizados en este estudio porque es ampliamente utilizado por la comunidad científica en campos como PLN, IA y Macro datos (*Big Data*), y cuenta con una amplia comunidad de soporte.

Se utilizaron herramientas como numpy, scikit-learn, pandas y matplotlib. Para garantizar la escalabilidad y el uso constante, se utilizó Apache Spark versión 3.0.1 por su capacidad de procesar grandes volúmenes de datos de manera distribuida y escalable.

En la Figura 2 se muestra el proceso del flujo de trabajo que se siguió para generar un modelo en el que sus partes más importantes en el proceso son el Paso 3 que se refiere a los métodos en el cual se transforman los textos a espacios vectoriales y el Paso 5 donde se forman los grupos con técnicas de agrupamiento midiendo las distancias geométricas de los vectores producidos desde el Paso 3.

El proceso ilustrado en la Figura 2 abarca la combinación de todas las técnicas que se incluyen en cada paso del proceso. Se produjeron 4320 modelos que se ejecutaron una sola vez cada uno y de los cuales es desprenden los resultados.

2.3. Carga de los datos y de los casos de prueba

Ya con los artículos pre-procesados, para cumplir con los criterios de tamaño máximo se procede a cargarlos en la memoria junto con los casos de prueba para que sean procesados, transformados y agrupados al mismo tiempo.

2.4. Pre-procesamiento

El siguiente paso es el pre-procesamiento, aquí se intercambian todas las letras mayúsculas por minúsculas seguido por una etapa de limpieza en la que se eliminan las palabras de paro (*stop words*), que son palabras que se denomina que no agregan valor al sentido del texto, esto facilita la transformación de texto en vectores numéricos y su manipulación en pasos subsecuentes.

Después de la limpieza de los artículos, sus textos se transforman en listas de palabras, a esto se le conoce como simbolización (*tokenization*) pues a cada palabra en la lista se le denomina símbolo (*token*).

A continuación se genera una serie de N-gramas de 2 y 3 palabras, esto con la intención de que las secuencias de palabras que sean más frecuentes y que usualmente representan conceptos de 2 o tres palabras, puedan ser capturadas también como un solo término.

2.5. Representación de textos en espacios vectoriales

Después que los artículos se convierten en símbolos, pueden ser procesados en la siguiente fase del modelo, en la cual incluimos cuatro aproximaciones recientemente utilizadas en la literatura. Estos métodos transforman las listas de símbolos contenidas en cada artículo en una representación de vectores numéricos que se mencionan a continuación:

CountVectorizer. El modelo *count vectorizer* (también llamado vectorizador de la frecuencia de los términos (*term frequency vectorizer*) o TF), es un modelo basado en la técnica de representación bolsa de palabras (*bag-of-words*), lo que implica que no se tiene en cuenta la información sobre la posición de los símbolos ni su contexto. En lugar de ello, se enfoca únicamente en la presencia y frecuencia de cada palabra en la colección de documentos.

En otras palabras registra el número de veces que aparece cada palabra de un vocabulario en un documento. Por esta razón, *countVectorizer* es especialmente útil en tareas de PLN donde el contexto y la posición de los símbolos no son esenciales para la tarea en cuestión, como la detección de similitudes entre documentos o la agrupación de documentos por temas [1].

En este modelo, el parámetro utilizado es el ”tamaño del vocabulario”(*vocab size*), que se refiere al número máximo de palabras que el modelo *CountVectorizer* crea en su vocabulario. Este vocabulario se forma a partir de los términos más importantes, ordenados por frecuencia de aparición en todo el corpus de texto, y se limita al tamaño especificado por el parámetro *vocab_size*. En este estudio se limita la búsqueda de modelos a la variación del parámetro *vocab_size* de tamaño 200, 400 y 800.

TF_IDF. TFIDF es un acrónimo que hace referencia a “frecuencia del término por su inverso en la frecuencia de los documentos” por sus siglas en inglés (*Term Frequency-Inverse Document Frequency*).

Esta técnica se utiliza para establecer una proporción entre la frecuencia de un término en un documento específico y su frecuencia en todo el corpus de documentos. De esta manera, si un término aparece con frecuencia en un documento pero no en los demás, su valor TFIDF será alto, lo que indica que es un término relevante y distintivo para ese documento.

El objetivo de la técnica TFIDF es identificar los términos clave de un documento en relación con el corpus completo. Se espera que los términos clave aparezcan con una frecuencia mayor en el documento en cuestión y con una frecuencia menor en el corpus. De esta forma, los términos que aparecen con mayor frecuencia en el documento tienen un valor TFIDF más alto y se consideran más importantes que los términos que aparecen con frecuencia similar en todos los documentos [13].

La técnica TFIDF es muy útil en el procesamiento de lenguaje natural en la clasificación de textos, ya que ayuda a identificar los términos más relevantes en un conjunto de documentos.

El parámetro *numFeatures* en este modelo determina la dimensión del vector de características. Un valor más alto para *numFeatures* dará como resultado un vector de características de mayor dimensión, lo que potencialmente capturará información más detallada de los datos de texto, pero también aumentará la complejidad computacional y el uso de la memoria.

Por otro lado, un valor más bajo para *numFeatures* dará como resultado un vector de características de menor dimensión, lo que podría perder parte de la información de los datos de texto pero también reducir la sobrecarga computacional. En este estudio se limita la búsqueda de modelos a la variación del parámetro *numFeatures* de tamaño 200, 400 y 800.

Word2vec. Word2vec es un conjunto de modelos de redes neuronales que se caracterizan por ser poco profundos (bolsa de palabras continua (*Continuous Bag of Words* (CBOW) y salta grama *Skip-Gram*). Las palabras que comparten contextos comunes en el corpus están ubicadas cerca unas de otras en el espacio vectorial. Esto significa que las palabras que tienen significados similares se encuentran en la misma zona del espacio vectorial [7].

Es importante destacar que los modelos de Word2vec no solo tienen en cuenta la frecuencia de las palabras en el corpus, sino que también consideran su contexto. De esta manera, las palabras que aparecen juntas con frecuencia en el corpus son codificadas en vectores cercanos en el espacio vectorial, lo que permite que las similitudes semánticas se reflejen en la ubicación de las palabras en el espacio.

El parámetro *vector_size* en el modelo Word2Vec determina el número de dimensiones en los vectores de palabras aprendidos. Un valor más alto para *vector_size* dará como resultado vectores de palabras de mayor dimensión, capturando potencialmente relaciones semánticas más matizadas entre palabras, pero también aumentando la complejidad computacional y el uso de memoria.

Por otro lado, un valor más bajo para *vector_size* dará como resultado vectores de palabras de dimensiones más bajas, lo que podría reducir la capacidad de capturar información semántica detallada pero también reducir la sobrecarga computacional.En este estudio se limita la búsqueda de modelos a la variación del parámetro *vector_size* de tamaño 200, 400 y 800.

BERT Se usó el modelo BETO, el cual es una variante del modelo BERT (*Bidirectional Encoder Representations from Transformers*) o Representación de Codificador Bidireccional desde Transformadores entrenado con un gran corpus en español.

BETO es de un tamaño similar al modelo BERT-base y fue entrenado con la técnica de enmascaramiento de todas las palabras (*Whole Word Masking technique* WWM) que toma en cuenta todas las sub palabras como prefijos o sufijos o variantes de la misma raíz como una sola palabra para que el significado original se mantenga durante todo el entrenamiento [3, 2].

En este estudio se limita la búsqueda de modelos a la variación del parámetro *max_seq_length* de tamaño 500, el limite máximo para este parámetro es de 516. El parámetro *max_seq_length* establece un límite superior en el número de símbolos que pueden estar presentes en una secuencia de entrada. Si una secuencia excede este límite, debe truncarse u omitirse para ajustarse a la longitud especificada.

Las secuencias que son más cortas que la longitud máxima especificada generalmente se rellenan con símbolos especiales para garantizar la uniformidad en los datos de entrada. Secuencias más largas pueden contener más contexto y proporcionar información más rica para el modelo, pero también aumentan la complejidad computacional y el uso de la memoria. Por otro lado, las secuencias más cortas pueden provocar la pérdida de información contextual importante.

2.6. Escalación y normalización

Una vez que los artículos fueron transformados a sus representaciones en espacios vectoriales, mediante los métodos ya mencionados, el siguiente paso consistió en seleccionar un método de escalado o normalización de entre los cuales se probaron escala mínima-máxima, estandarización y ninguno.

2.7. Agrupamiento

El siguiente paso consistió en formar los grupos y para cada uno de ellos se realizó la búsqueda dentro de los siguientes parámetros: $numb\_clusters\_s = [20, 22, 24, 26, 28, 30, 32, 34, 36]$. Se eligió este rango en función de la cantidad de temas presentes en la CPEUM.

La CPEUM consta de 136 artículos distribuidos en nueve títulos, estos títulos a su vez se dividen en capítulos, sumando 16 secciones en total, más aquellos que deben ser clasificados como derogados y otras categorías que pudieran estar presentes se eligieron 20 grupos como mínimo para empezar la búsqueda de los modelos.

Para realizar el agrupamiento de los artículos en su representación vectorial, se utilizaron los siguientes métodos:

Distribuciones gaussianas mixtas. El método de Distribuciones gaussianas mixtas (*Gaussian Mixture Models* GMM) en cuestión no es compatible con el proceso de estandarización llevado a cabo en el paso anterior, ya que está diseñado para trabajar con las distribuciones de los espacios vectoriales de los artículos, y la estandarización provoca una pérdida de la información fundamental con la que opera este algoritmo, por esta razón solo se corrieron modelos con escalación de datos máxima y mínima y modelos sin ningún tipo de escalación o normalización.

El parámetro "tipos de covarianza" o *cov_types* en un modelo GMM especifica el tipo de matriz de covarianza que se utiliza para modelar la estructura de covarianza de las distribuciones gaussianas en la mezcla. La matriz de covarianza representa la covarianza o correlación entre diferentes características o dimensiones de los datos. para este método, se utilizó el parámetro *cov_types* de tipo *full*, *diag* y *spherical*.

El *cov_types* de tipo completo (*full*) asume que cada componente gaussiano en la mezcla tiene su propia matriz de covarianza completa, que puede capturar correlaciones arbitrarias entre diferentes características o dimensiones de los datos. Sin embargo, esto puede ser computacional-mente costoso y puede requerir una gran cantidad de parámetros para estimar.

El *cov_types* de tipo *diag* asume que cada componente gaussiano en la mezcla tiene su propia matriz de covarianza diagonal, que solo modela las variaciones de las características individuales sin capturar ninguna correlación entre ellas. Esto puede ser computacional-mente eficiente y puede funcionar bien para datos con poca o ninguna correlación entre características.

El *cov_types* de tipo esférico (*spherical*) asume que cada componente gaussiano en la mezcla tiene su propia matriz de covarianza esférica, que es una matriz de identidad escalada. Esto significa que todas las características tienen la misma varianza dentro de cada componente, sin capturar ninguna correlación entre ellas. Esto puede ser útil para datos con varianzas isotrópicas o similares en todas las entidades.

Agrupamiento aglomerante. Para el modelo de agrupamiento aglomerante *Agglomerative clustering* se utilizaron los parámetros de afinidad (*affinity*) y de enlace (*linkage*); para el parámetro de afinidad se utilizó la métrica euclidiana y para el método de enlace o encadenamiento se utilizó la estrategia de aglomeración de pabellón (*ward*). El parámetro de "afinidad" *aff* en un modelo de agrupamiento aglomerante determina el método utilizado para calcular las distancias por pares o las similitudes entre los puntos de datos.

Euclidean calcula la distancia euclidiana entre pares de puntos de datos, que es la distancia en línea recta entre dos puntos en el espacio euclidiano. Es la opción predeterminada en muchos algoritmos de agrupamiento aglomerativo y es adecuada para datos con características numéricas continuas.

El parámetro de "vinculación" o enlace en un modelo de agrupamiento aglomerativo especifica el método utilizado para calcular la diferencia o similitud por pares entre los grupos al fusionarlos. El método *ward* calcula el aumento en la suma de las diferencias al cuadrado (SSD) dentro de los grupos al fusionarlos. Tiende a producir grupos más compactos y esféricos, y es adecuado para datos con características numéricas continuas.

K medias. Para este método utiliza el parámetro *numIterations_s* con un valor de 100. El algoritmo K medias comienza inicializando aleatoriamente K centroides, donde K es el número de grupos especificado. Luego, alterna entre dos pasos:

Paso 1: Asignación de cada punto de datos al centroide más cercano en función de la distancia euclidiana u otra métrica de distancia.

Paso 2: Actualización de los centroides calculando la media de todos los puntos de datos asignados a cada centroide.

Estos dos pasos se repiten iterativamente hasta que se cumple un criterio de convergencia, que es el número máximo de iteraciones o un umbral de tolerancia en el cambio d e centroides. El parámetro "número de iteraciones" o *numIteration_s* en un modelo de agrupamiento de K-medias determina el número máximo de veces que el algoritmo realizará los pasos de asignación y actualización antes de detenerse, independientemente de si los centroides han convergido o no. Si el algoritmo alcanza el número máximo de iteraciones antes de converger, se detendrá y devolverá las asignaciones de grupo y los centroides actuales como resultado final.

2.8. Evaluación

Elaboración de los casos de pruebas

Los casos de prueba se obtuvieron de un experto en la materia legal y de un Tesauro jurídico de la SCJN. Son conjuntos de palabras que se sustituyen fácilmente a lo largo de todas las constituciones y que no cambian el significado de los artículos que los contienen.

Así el caso de prueba 1 hace referencia a grupos originarios; grupo étnico; grupos étnicos; indígenas; pueblos indígenas; pueblos tribales; pueblos indios; Poblaciones Indígenas y Tribales, etc... El caso de prueba 2 a la libre determinación; autonomía; auto regulación; etc... El caso de prueba 3 a la Acta Constitucional; Carta constitucional; Carta federal; Carta magna; Código fundamental; etc... Y el caso de Prueba 4 a Derechos de la persona humana; Derechos del hombre; Derechos esenciales del hombre; Derechos implícitos; etc...

Métrica de evaluación

Para evaluar los modelos, se utilizó una métrica especializada (*ad-hoc*) a la que se denominó "*recall* integrado".

El funcionamiento de esta métrica requiere saber qué artículos contienen una o varias palabras del caso de prueba a las que definimos como palabras clave.

Se escogió una variación de la métrica "*recall*" porque solo sabemos cuales artículos contienen palabras clave, por lo que los elementos relevantes son los verdaderos positivos (*True Positives* TP) que representan el grupo que contienen el caso de prueba y los artículos que tienen por lo menos una de las palabras clave y los falsos negativos (*False Negatives* FN) que representan los grupos que contienen artículos con alguna de las palabras clave pero no el caso de prueba.

Para lograrlo se resolvieron los siguientes pasos en orden:

Paso 1: En este paso se agregó una columna con una bandera que nos ayudó a identificar los artículos que contenían por lo menos una de las palabras clave y comparar y verificar que por lo menos los textos que contenían estas palabras pudieran ser agrupados en el mismo grupo y los que no se agrupaban donde mismo poder encontrar las diferencias en el contexto del artículo.

Paso 2: Se utilizó el artículo personalizado que contenía las palabras clave del caso de prueba para determinar el agrupamiento correcto al que debían pertenecer los artículos semánticamente similares y se descartaron los demás grupos.

Paso 3: Se compararon los resultados de los pasos 1 y 2 y se obtuvieron las dos categorías que se necesitan para calcular la métrica de evaluación a partir de los resultados de los modelos de agrupación verdaderos positivos TP y falsos negativos FN.

Paso 4: Se separaron los modelos según sus características de composición similares como sigue:

1. método de representación de texto en espacios vectoriales y
2. método de agrupamiento.

Paso 5: Se utilizaron los valores de TP y FN para calcular la métrica especializada "*recall* integrado" sobre los modelos que difieren en el caso de prueba.

La función "*recall* integrado" se calculó de acuerdo a la siguiente ecuación 1:

$$recall\_integrado = \frac{\sum_i^n tp_i}{\sum_i^n tp_i + \sum_i^n fn_i}, \tag{1}$$

donde:

$\sum_i^n tp_i$ = es la sumatoria de todos los verdaderos positivos
$\sum_i^n fn_i$ = es la sumatoria de todos los falsos negativos

Ambas sumatorias se calcularon sobre los modelos similares en composición, esto es, los que se componen por el mismo algoritmo de representación y de agrupamiento, incluyendo todos los modelos que utilizan cada uno de los casos de prueba.

2.9. Visualización

Por último se graficaron los resultados de la evaluación con la métrica anterior en cada punto contra el parámetro cantidad de grupos utilizado, y se mostraron las distribuciones que se obtuvieron por cada modelo entrenado.

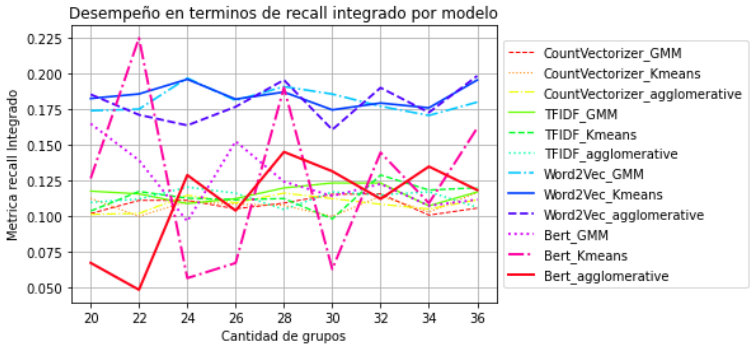

Fig. 3. Desempeño de los modelos en términos de "*recall* integrado".

3. Resultados

El desempeño de cada modelo se puede observar en la figura 3, donde los modelos con mejores comportamientos en general son los que se entrenaron con un representación en espacios vectoriales *Word2vec* sin embargo el modelo que tuvo el mejor desempeño individual fue el entrenado con un representación en espacios vectoriales BETO y con el modelo de agrupamiento de distribuciones gaussianas mixtas.

En la figura 4 se puede observar mejor las distribuciones de los comportamientos de los modelos, agrupados por el algoritmo de representación en espacios vectoriales utilizado, donde los modelos relacionados al *count vectorizer* en promedio tuvieron un desempeño de 0.110 , la representación tf idf tuvo en promedio un desempeño de 0.120, la representación BERT en promedio 0.124 y la representación word2vec 0.180 con respecto a la métrica "*recall* integrado". De éstos números se deriva que los modelos de representación que se calculan a partir de redes neuronales, es decir, BERT y *word2vec*, son los que obtienen un mejor desempeño.

A pesar que el modelo que obtuvo individualmente el mejor resultado utiliza BERT para la representación en el espacio vectorial, la dispersión de los resultados obtenidos por el resto de los modelos que también utilizan BERT es la mayor de todos los algoritmos de representación utilizados, esto significa que el desempeño obtenido no es consistente.

Se considera que los modelos que utilizan word2vec como algoritmo de representación tuvieron en general una baja dispersión y un mejor desempeño que los demás en términos de recall integrado", por esta razón se concluye que este algoritmo es más consistente que el resto de los modelos comparados en este trabajo.

4. Trabajo futuro

Se cree que los resultados obtenidos pueden ser aplicables a otros corpus de normas, reglamentos o materias legales y este trabajo busca establecer un precedente en el campo ya que, hasta donde sabemos, no existen estudios similares en la literatura en el ámbito legal.

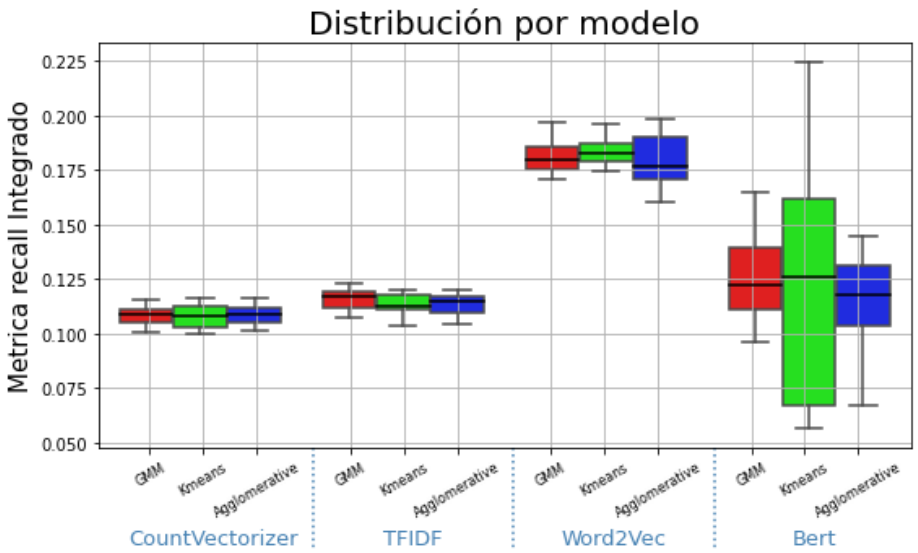

Fig. 4. Distribución de los modelos en términos de "*recall* integrado".

Es importante tener en cuenta que el alcance de este trabajo se limitó a la selección de los métodos previamente mostrados, debido a su uso demostrado en trabajos anteriores con problemáticas similares y a la complejidad y costo computacional de implementar modelos con una mayor cantidad de parámetros.

Este trabajo se puede extender en el futuro como una aplicación que realice una búsqueda por término para obtener los artículos específicos que se relacionan al mismo, también se pude extender y mejorar la cantidad de casos de prueba, además de evaluar los resultados de los métodos de agrupación con métricas como el índice de Calinski-Harabasz o el índice de Davies-Bouldin que miden la cercanía entre los grupos generados.

Se pueden explorar otras métricas de similitud para agrupamiento de los textos. Otra posibilidad es ejecutar varias veces todos los modelos y realizar los cálculos de nivel de significancia de los resultados de los agrupamientos.

Una limitación de la aplicación de estos modelos es la dependencia de un experto en el área legal para la generación de casos de prueba y para la interpretación de los resultados debido a la gran cantidad de casos posibles de los cuales solo se utilizaron cuatro en este trabajo para la métrica de evaluación, por esta razón se propone el desarrollo modelos de aprendizaje de tipo no supervisado.

5. Conclusiones

En este estudio, se desarrollaron modelos que identifican textos semánticamente similares utilizando técnicas de procesamiento del lenguaje natural (PLN) para ayudar en el problema del derecho comparado.

Se probaron cuatro técnicas de representación en espacios vectoriales (count vectorizer, TF-IDF, word2vec y BERT) combinadas con tres métodos de agrupamiento (GMM, agrupamiento aglomerante y k-medias).

Se aplicaron estas técnicas a un corpus legal compuesto por las constituciones de todos los estados y la constitución federal, y se curó de forma semi-automática. El corpus resultante consta de 14047 documentos.

Para evaluar los resultados, se utilizaron casos de prueba de diferentes términos legales semánticamente similares y una métrica ad-hoc llamada recall integrado". El modelo individual obtuvo el mejor resultado fue el formado por la técnica de representación en espacios vectoriales BERT en combinación con el método de agrupamiento k-medias.

Sin embargo, debido a la dispersión que tuvieron los otros modelos que utilizaron BERT como técnica de representación no se escoge como el algoritmo con mejor desempeño por su inconsistencia.

No obstante, Los modelos formados por la técnica de representación en espacios vectoriales word2vec tuvieron una menor dispersión y en promedio los mejores resultados independientemente del método de agrupamiento utilizado.

Referencias

1. Basarkar, A.: Document classification using machine learning. Ph. D. thesis, San Jose State University, pp. 1–56 (2017) doi: 10.31979/etd.6jmu-9xdt
2. Cañete, J., Chaperon, G., Fuentes, R., Ho, J. H., Kang, H., Pérez, J.: Spanish pre-trained BERT model and evaluation data. In: PML4DC at ICLR 2020 (2020)
3. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186 (2019) doi: 10.18653/v1/n19-1423
4. Fix-Fierro, H.: ¿Por qué se reforma tanto la constitución mexicana de 1917? Hacia la renovación del texto y la cultura de la constitución. Cien Ensayos para el Centenario, vol. 4, pp. 143–162 (2017)
5. Flores-Sánchez, J. A., Maglioni-Montalvo, R.: Derecho comparado en investigación y enseñanza de derecho en carreras económico administrativas. Hitos de Ciencias Económico Administrativas, vol. 25, no. 71, pp. 136–147 (2020) doi: 10.19136/hitos.a25n71.3608
6. Katz, D. M., Hartung, D., Gerlach, L., Jana, A., Bommarito, M. J.: Natural language processing in the legal domain. SSRN, pp. 1–13 (2023) doi: 10.2139/ssrn.4336224
7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR (Workshop Poster) (2013) doi: 10.48550/arXiv.1301.3781
8. Örn Arnarsson, I., Frost, O., Gustavsson, E., Jirstrand, M., Malmqvist, J.: Natural language processing methods for knowledge management—applying document clustering for fast search and grouping of engineering documents. Concurrent Engineering Research and Applications, vol. 29, no. 3, pp. 142–152 (2021) doi: 10.1177/1063293x20982973
9. Risco, J.: Los malditos vacíos legales. El financiero (2017)
10. Romero-Pérez, J. E.: Notas sobre la interpretación jurídica. Revista de Ciencias Jurídicas, vol. 1, no. 133, pp. 79–102 (2014)
11. Secretaría de Servicios Parlamentarios: Leyes federales vigentes (2020) www.diputados.gob.mx/LeyesBiblio/index.htm

12. Shahmirzadi, O., Lugowski, A., Younge, K.: Text similarity in vector space models: A comparative study. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 659–666 (2018) doi: 10.1109/ICMLA.2019.00120
13. Siregar, A. M., Faisal, S., Tukino, A. P., Simarangkir, M. S.: Comparison study of term weighting optimally with svm in sentiment analysis. In: Proceedings of The 2nd International Conference On Advance And Scientific Innovation (2019) doi: 10.4108/eai.18-7-2019.2288508