



**Evaluación de Técnicas de Clustering para agrupar
Términos del Lenguaje del Dominio capturados a través del
Glosario LEL**

Alumno: Germán Adolfo Nuñez.

Profesora: Marcela Rosalba Samela.

Tutor Técnico: Dr. Leandro Antonelli.

**Trabajo Final para ser presentado para obtener el Título en
Carrera de Licenciatura en Gestión de la Tecnología Informática.**

Julio 2026

Resumen

En este trabajo se propuso la evaluación de técnicas y herramientas basadas en Inteligencia Artificial (IA) orientadas al tratamiento de especificaciones técnicas en entornos documentales complejos, específicamente técnicas de clusterización. El objetivo principal fue establecer si era posible el uso de dichas técnicas y herramientas para analizar y agrupar términos en base a un documento de ingeniería de requerimientos y definiendo también la mejor técnica y herramienta. Se probaron modelos capaces de agrupar términos de un glosario de Léxico Extendido del Lenguaje (LEL), abordando desafíos como la ambigüedad y la redundancia inherentes a la vasta cantidad de estos que podría haber en dominios muy complejos. La metodología constó de pruebas y comparativas de distintos métodos de clusterización. Los resultados obtenidos demuestran si estas herramientas tienen el potencial para optimizar la toma de decisiones y gestión del conocimiento en organizaciones técnicas y multidisciplinarias.

Palabras clave: clustering, glosario lel, ingeniería de requerimientos, inteligencia artificial, léxico extendido del lenguaje, procesamiento del lenguaje natural

Abstract

This work proposed the evaluation of techniques and tools based on Artificial Intelligence (AI) aimed at the treatment of technical specifications in complex documentary environments, specifically clustering techniques. The main objective was to determine whether it was possible to use such techniques and tools to analyze and group terms based on a requirements engineering document, while also defining the best technique and tool. Models capable of grouping terms from a Language Extended Lexicon (LEL) glossary were tested, addressing challenges such as ambiguity and redundancy inherent in the vast number of terms that could exist in highly complex domains. The methodology consisted of tests and comparisons of different clustering methods. The results obtained demonstrate whether these tools have the potential to optimize decision-making and knowledge management in technical and multidisciplinary organizations.

Keywords: artificial intelligence, clustering, natural language processing, lel glossary, language extended lexicon, requirements engineering.

Agradecimientos

En este nuevo y gran desafío que realizo para el final de mi carrera, agradezco a aquellos que siempre estuvieron a mi lado mi madre y padre, Ana y Ramón, que me apoyaron en todo lo que pudieron ayudándome mucho no solo en la realización de este trabajo sino en el transcurso de toda mi carrera.

También agradezco al doctor Leandro Antonelli por brindando una luz guía en los momentos en que estaba perdido durante el desarrollo de la investigación.

Índice General

Resumen	1
Abstract.....	2
Agradecimientos	3
Índice de Tablas	7
Índice de Figuras	8
Capítulo 1 - Introducción General.....	9
1.1 Planteamiento del problema	9
1.2 Justificación.....	9
1.3 Preguntas de investigación	11
1.4 Hipótesis.....	11
1.5 Objetivo General.....	11
1.6 Objetivos específicos	11
1.7 Metodología de investigación.....	11
Capítulo 2 - Estado del Arte	15
2.1 El Desafío del Lenguaje Natural en la Ingeniería de Requerimientos.....	15
2.2 NLP4RE: Un Campo de Investigación Consolidado.....	16
2.3 Evolución de la representación vectorial: De TF-IDF a Embeddings.	17
2.3.1 Word Embeddings estáticos: Word2Vec y GloVe.....	18
2.3.2 Embeddings contextuales: BERT y Sentence-BERT	18
2.3.3 Embeddings generados por LLM modernos.....	18
2.4 Aplicación de NLP a la gestión de glosarios LEL	19
2.4.1 Extracción de conocimiento de los glosarios LEL a través de PLN	19
2.4.2 LLMs para la construcción y organización de dominio	19
2.4.3 LLMs para el agrupamiento semántico de términos	20
2.5 Clustering aplicado a Ingeniería de Requerimientos	21
2.5.1 Clustering sobre casos de uso	21
2.6 Síntesis y brecha identificada	22
Capítulo 3 - Marco teórico	25
3.1 Ingeniería de Requerimientos.....	25
3.1.1 Requerimientos	25

3.1.2 Tipos de requerimientos	25
3.1.3 Importancia de ingeniería de requerimientos (IR)	26
3.1.4 Glosarios de dominio	27
3.2 Léxico Extendido del Lenguaje (LEL)	27
3.2.1 Estructura de los Glosarios LEL	27
3.3 Inteligencia Artificial y Procesamiento de Lenguaje Natural	31
3.3.1 Fundamentos del PLN	31
3.3.2 Tareas fundamentales PLN	32
3.3.3 Representación vectorial de texto (embedding).....	33
3.3.4 Técnicas de Prompting	36
Capítulo 4 – Contribución	38
4.1 Clustering	38
4.1.1 Tipos de Clustering	39
4.1.2 Técnicas de clustering	39
4.1.3 Criterios para determinar número de clústeres	45
4.1.4 Métricas de evaluación internas de clustering	46
4.2 Determinación de numero de clústeres en experimento	50
4.3 Configuración final del experimento	52
4.4 Resultados obtenidos.....	52
4.4.1 Métricas Internas.....	53
4.4.2 Métricas externas	54
4.5 Comparación entre representaciones	55
4.6 Comparación entre algoritmos.....	56
4.7 Análisis de la matriz de contingencia.....	57
Capítulo 5 - Discusión	59
5.1 Respuesta a la pregunta de investigación 1	59
5.2 Respuesta a la pregunta de investigación 2	60
5.3 Evaluación de la hipótesis	61
5.4 Limitaciones	62
Capítulo 6 - Conclusiones Generales	64

6.1 Trabajo Futuro	64
Referencias	66
Anexos	70
Anexo A. Glosario de términos técnicos.....	70
Anexo B. Prompts utilizados en el experimento con el LLM.....	72
B.1 Reglas comunes a ambas estrategias	72
B.2 Prompt de la estrategia zero-shot	72
B.3 Prompt de la estrategia few-shot	73
Anexo C. Conjunto de referencia	74
Anexo D. Barrido de parámetros de DBSCAN.....	77

Índice de Tablas

Tabla 1 - Caracterización del glosario LEL de estudio	12
Tabla 2 - Descripción de un sujeto del glosario LEL	29
Tabla 3 - Descripción de un objeto del glosario LEL	30
Tabla 4 - Descripción de un verbo del glosario LEL	31
Tabla 5 - Descripción de un estado del glosario LEL	31
Tabla 6 - Resultados de métricas internas de las distintas combinaciones	53
Tabla 7 - Resultados de métricas externas contra el conjunto de referencia	54
Tabla 8 - Matriz de contingencia: SBERT + K-Means vs. referencia	57
Tabla 9 - Matriz de contingencia: LLM (zero-shot) vs Referencia	61
Tabla C1 - Símbolos del glosario LEL y subdominio asignado	74
Tabla D1 - Mejores configuraciones de DBSCAN por representación	77

Índice de Figuras

Figura 1 - Método del codo y silueta promedio calculada para TF-IDF	51
--	-----------

Capítulo 1 - Introducción General

1.1 Planteamiento del problema

En el ciclo de vida del desarrollo de una herramienta de software, la ingeniería de requerimientos es una de las primeras etapas. Esta posee una relevancia muy considerable debido a que errores cometidos en esta etapa requieren un retrabajo cuyo costo es muy alto si no se detectan a tiempo.

Para mitigar este problema se crearon las metodologías ágiles las cuales con un desarrollo incremental e iterativo basado en el desarrollo de entregables funcionales donde se validan constantemente los resultados. En algunos casos el dominio es sencillo y eso permite aplicar la metodología directamente, sin embargo, en otros casos son más complejos lo que obliga a poseer conocimiento del dominio para poder aplicar la metodología. Y existen veces en donde el dominio es tan complejo que aplicar la metodología ágil no es posible debido al vasto conocimiento del dominio que es necesario tener. Esto puede derivar en problemas de malentendidos, por parte de los desarrolladores y testers, de los requerimientos del sistema por falta de conocimiento del dominio, provocando que se cometan errores potencialmente críticos difíciles de solucionar, como ya mencionamos anteriormente.

Para la comprensión del dominio es necesario comprender el lenguaje. En la ingeniería de requerimientos es común el uso de glosarios. En particular, hay un glosario que se denomina glosario LEL. En dominios muy complejos la cantidad de términos en este glosario puede ser muy alta, y una forma de lidiar con esta complejidad es agrupando términos según un criterio. Es posible agrupar el conocimiento a partir de subdominios.

La Inteligencia Artificial (IA) ha hecho grandes avances en el procesamiento del lenguaje natural y los LLMs a través del machine learning. Posibilitando el tratamiento de textos con diferentes finalidades, entre ellas la de clusterizar (agrupar), identificar similitudes, extraer información, etc.

1.2 Justificación

La aplicación de técnicas de clusterización en la fase de ingeniería de requerimientos es crucial debido a que esta ayuda a la claridad, optimización y a estructurar la información. Como lo menciona Arora et al. (2017) en su trabajo sobre la extracción y agrupamiento de términos de glosarios de requerimientos en IR, un glosario constituye una parte muy importante en cualquier documento de requerimientos debido a que cuando se hacen explícitos los

términos de un dominio y se brinda información sobre ellos, lo que genera es la disminución de imprecisión y la ambigüedad que se generarían si se usara lenguaje natural.

Como primera virtud del clustering aplicado a glosarios LEL podemos mencionar la mejora en la coherencia y la consistencia de las terminologías. Como lo indican Arora et al. (2017) los documentos de requerimientos son propensos a incluir variaciones no intencionales ya sea por la flexibilidad del lenguaje natural, las diferencias de terminología o el estilo de redacción entre los distintos individuos u organizaciones. Es por esta razón que es importante poner en conocimiento de estas variaciones a los analistas. Aquí es donde entra la virtud del clustering debido a que, cuando un analista inspecciona un clúster puede asegurar de que no exista sinónimos desconocidos o no deseados entre los términos agrupados.

La segunda virtud se basa en la facilidad de navegación y comprensión del dominio. Aquí Arora et al. (2017) destacan que los clústeres formados representan agrupaciones de términos que están relacionados y esto ayuda a los analistas en su labor, esto incluye tanto la decisión sobre qué términos se incluirán, la redacción de definiciones y la detección temprana de potenciales problemas de consistencia. Esta necesidad adquiere particular importancia cuando tenemos a autores como Roldán Valdiviezo (2025) quien destaca que los glosarios LEL, suelen tener distintos tipos de redacción que pueden afectar la comprensión del lenguaje utilizado.

La tercera virtud es que el clustering ayuda a aumentar la calidad de las definiciones, y optimiza la gestión del glosario. Sobre este tema Arora et al. (2017) mencionan que el contexto que provee el clúster, comparándolo a si solo tuvieran términos aislados, ayuda a los analistas a redactar definiciones más precisas sobre los términos del glosario. Esto se traduciría en menos ambigüedad en los requerimientos descritos, y por consiguiente menos malinterpretaciones de parte de los desarrolladores y testers. En esta misma línea el trabajo de Delle Ville et al. (2024) demuestra que el análisis automatizado de los artefactos de IR permite a los analistas detectar redundancias o similares que de otra forma pasarían desapercibidas.

Además de lo anteriormente expuesto también podemos mencionar que el clustering sobre términos de glosario también cuenta con respaldo empírico. En el estudio de Arora et al. (2017) se realizaron varias encuestas sobre los resultados en los casos estudiados, el resultado fue que los expertos estuvieron de acuerdo en el 89,6% de los casos (60 sobre 67) en que los clústeres resultaban útiles en la detección de términos relacionados. En el 88% de los casos que ayudaba definir los términos con mayor precisión y en el 61,2% de los casos en que facilitaba la identificación de variaciones en la terminología.

Por todo lo anterior expuesto podemos decir que la clusterización no solo organiza el léxico, sino que hace que transforma el glosario LEL para que sea algo más que una simple lista de definiciones, esta se vuelve una estructura más rica y accionable.

1.3 Preguntas de investigación

¿Se pueden utilizar las técnicas de clustering para identificar los distintos subdominios en los glosarios LEL?

¿En caso de ser posible, cuál es la combinación más efectiva para realizarlo?

1.4 Hipótesis

La aplicación de técnicas de clusterización avanzadas a los términos de un glosario LEL permite una agrupación semántica para identificar distintos subdominios dentro del dominio de dicho glosario.

1.5 Objetivo General

Este trabajo tiene por objetivo evaluar técnicas de clusterización para el agrupamiento de términos existentes en el glosario LEL, con el fin de saber si es posible y conveniente hacerlo mediante las técnicas de clusterización y en caso de ser posible establecer la forma más efectiva.

1.6 Objetivos específicos

Para cumplir con el objetivo general se proponen los siguientes objetivos específicos:

1. Relevar las técnicas existentes de clusterización.
2. Preparar los términos del glosario LEL para su procesamiento mediante técnicas de clustering.
3. Evaluar las combinaciones de representaciones vectoriales (TF-IDF y SBERT) con algoritmos de clusterización, así como el uso de LLMs a través del diseño de prompts.
4. Analizar los resultados obtenidos.

1.7 Metodología de investigación

En la presente investigación se adopta un enfoque de investigación cuantitativo, de tipo experimental-comparativo con naturaleza exploratoria. Se eligió un enfoque cuantitativo debido a que todo el procesamiento se sustenta en la transformación del texto del glosario a representaciones numéricas y en la medición del resultado mediante métricas. En cuanto a la

parte experimental-comparativa se contrastan distintas combinaciones de técnicas de representación vectorial y algoritmos de agrupamiento. Por último, se opta por una naturaleza exploratoria dado que el clustering es una técnica no supervisada, sin embargo a diferencia de un escenario totalmente no supervisado, en esta investigación se construye un conjunto de referencia para poder aplicar las métricas externas, que no interviene en el proceso del agrupamiento, sino que se utiliza únicamente en la etapa de análisis de resultados para contrastar en qué medida coinciden estos conjuntos de referencia con los agrupados por los distintos algoritmos.

Corpus de estudio

El corpus está conformado por un glosario LEL real el cual corresponde a un sistema de gestión de laboratorio, del cual ya se presentó un símbolo de ejemplo en el marco teórico. El glosario contiene 78 símbolos, distribuidos en las cuatro categorías propias del LEL: 5 sujetos, 16 objetos, 37 verbos y 20 estados. En una inspección preliminar se puede observar que el glosario posee varias agrupaciones temáticas, lo que confirma que este cuenta con una estructura de subdominios susceptible a ser descubierta por los algoritmos de clusterización.

Para cada símbolo LEL, el cual se utiliza como unidad de análisis, se construye un documento textual concatenando su nombre, su noción y su impacto, esto es porque es en estos atributos donde se encuentra la carga semántica que permite distinguir un subdominio de otro.

Tabla 1

Caracterización del glosario LEL de estudio

Característica	Glosario (Laboratorio de análisis clínicos)
Dominio de aplicación	Gestión de laboratorio de análisis clínicos
N.º total de símbolos	78
Sujetos	5
Objetos	16
Verbos	37
Estados	20
N.º de subdominios (referencia)	8

Nota. Elaboración propia: Esta tabla describe tanto el dominio como la cantidad de símbolos que posee el documento LEL a procesar.

Construcción del conjunto de referencia

Dada la naturaleza no supervisada del clustering, las métricas internas solo permiten medir la cohesión y separación de los clústeres formados, pero no si estos son representativos de los subdominios reales del glosario. Para poder responder la pregunta de investigación se construirá un conjunto de referencia, que consiste en la asignación manual de cada símbolo a un subdominio. Esta asignación será realizada por el autor de este trabajo a partir del conocimiento del dominio y, en medida de lo posible, validada por el tutor experto. Conviene recordar que, como señalan Alonso et al. (2025), en clustering no existe un valor verdadero objetivo contra el cual comparar como ocurre en los métodos supervisados, por eso este conjunto se entiende como una referencia y no como una verdad irrefutable.

Algoritmos de clustering

En base a la brecha que se identifica en el estado del arte, se evalúan cinco algoritmos de clusterización los cuales son descritos en el apartado del marco teórico. Estos son K-Means (basado en centroides), agrupamiento jerárquico aglomerativo (basado en conectividad), Gaussian Mixture Model (basado en distribución), DBSCAN (basado en densidad) y Affinity Propagation (basado en intercambio de mensajes).

Métricas de evaluación

La evaluación combina métricas tanto internas como externas junto a un análisis cualitativo. Cubriendo así, por un lado, la calidad estructural del agrupamiento con las métricas internas y por el otro la similitud que tiene el agrupamiento obtenido y el conjunto de referencia es analizada con las métricas externas. Para las métricas externas se opta por dos índices corregidos por azar: Adjusted Rand Index (ARI), que razona sobre pares de elementos, y la Adjusted Mutual Information (AMI), que razona sobre la información compartida entre ambas particiones. Se escogen estas versiones debido a que las originales, Rand Index (RI) y Normalized Mutual Information (NMI), no descuentan el acuerdo esperable por azar, lo que produce que tiendan a aumentar el número de clústeres, afectando en mayor medida a algoritmos como DBSCAN y Affinity Propagation.

Para las métricas internas se eligió trabajar con Coeficiente de Silueta, Índice de Caliński-Harabasz e Índice de Davies-Bouldin, esta elección es debido a que son las más utilizadas en la literatura relevada, y además estas tres métricas al estar basadas en criterios distintos pero complementarios permiten obtener una valoración más completa de lo que se obtendría usando solo una de las métricas, puesto que posibles limitaciones de una métrica quedan compensadas por las restantes.

Herramientas y entorno de trabajo

La implementación se realiza con el lenguaje de Python. Tanto para los algoritmos de clusterización y las métricas se utilizará la biblioteca de scikit-learn, en cuanto al preprocesamiento lingüístico en español se utiliza la herramienta spaCy, para la reducción de dimensionalidad: PCA, y por último la manipulación de los datos y visualización de resultados se usan las herramientas de pandas y NumPy. Para el experimento de agrupamiento mediante prompting se utiliza el modelo de Llama 3.1 ejecutado de forma local con Ollama.

Capítulo 2 - Estado del Arte

2.1 El Desafío del Lenguaje Natural en la Ingeniería de Requerimientos

La Ingeniería de Requerimientos (IR) es una de las fases más críticas en el ciclo de vida del desarrollo de software (Umar y Lano, 2024). Errores, omisiones o malentendidos en esta etapa inicial podrían propagarse a lo largo de todo el proyecto, resultando en sobrecostos significativos, retrasos en la entrega y, en el peor de los casos, en un producto que no satisface las necesidades de los stakeholders. Esto es apoyado por el trabajo realizado por Khalid et al. (2025) el cual sostiene que una mala gestión del tiempo, dedicándole menor tiempo a la etapa de IR puede desembocar en el fracaso del software que se quiere crear. El núcleo de este desafío reside en el principal vehículo de comunicación utilizado: el Lenguaje Natural (LN). A pesar de su expresividad y accesibilidad para todos los involucrados, el LN es inherentemente propenso a defectos como la ambigüedad, la inconsistencia, la vaguedad y la incompletitud. Una misma frase puede ser interpretada de múltiples maneras por diferentes personas, llevando a especificaciones de escaso contenido o contradictorias (Antonelli et al., 2024).

Para mitigar estos riesgos, la disciplina de la IR (Ingeniería de Requerimientos) ha desarrollado artefactos y metodologías que buscan imponer una estructura y un control sobre el lenguaje del dominio. Entre estos, los glosarios de dominio desempeñan un papel fundamental. Una implementación semi estructurada es el Léxico Extendido del Lenguaje (LEL), un glosario que según Roldán Valdiviezo (2025) “busca comprender el lenguaje de un dominio de aplicación específico sin enfocarse directamente en el software de la aplicación” (p.113). Sin embargo, esta solución introduce un nuevo desafío: en dominios de aplicaciones grandes o complejas, los glosarios LEL pueden crecer hasta contener cientos o incluso miles de términos, volviéndose extensos y difíciles de gestionar, navegar y comprender en su totalidad. Esta complejidad a gran escala puede dificultar la identificación de relaciones importantes entre conceptos y la identificación de subdominios o áreas funcionales cohesivas.

Es en este punto donde la aplicación de técnicas de Inteligencia Artificial (IA), y más específicamente del Procesamiento del Lenguaje Natural (NLP), emerge como una prometedora solución para automatizar el análisis, la estructuración y la gestión de estos artefactos de IR. La capacidad de las técnicas de NLP para procesar y "comprender" grandes volúmenes de texto ofrece una vía para gestionar la complejidad de los glosarios LEL, transformándolos de listas estáticas en estructuras de conocimiento dinámicas y navegables.

La intención de este estado del arte es analizar trabajos relacionados a la clusterización de glosarios LEL, comenzando desde lo más genérico (NLP4RE) hasta lo más específico que es la propia clusterización.

2.2 NLP4RE: Un Campo de Investigación Consolidado.

Para realizar una clusterización en textos, primero es importante transformar el texto en una representación numérica que capture el significado de este. La evolución a lo largo del tiempo de estas técnicas es fundamental para la comprensión de este estado del arte. El proceso de estas técnicas ha pasado desde el emparejamiento superficial de palabras clave hasta una comprensión contextual profunda del lenguaje.

Según un estudio de mapeo sistemático de la literatura de Zhao et al. (2021) las herramientas de “Procesamiento de Lenguaje Natural para la Ingeniería de Requerimientos”, NLP4RE para sus siglas en inglés las cuales utilizaremos desde ahora para simpleza de lectura, fueron diseñadas para analizar y facilitar la comprensión de las especificaciones del dominio de sistemas, expresadas en lenguaje natural. También ha proporcionado un marco conceptual robusto para el campo de NLP4RE. Ha clasificado la investigación existente, identificado las principales áreas de aplicación y trazado la evolución de las técnicas utilizadas a lo largo del tiempo. Las tareas clave abordadas dentro de NLP4RE incluyen, entre otras, la clasificación de requisitos (por ejemplo, funcionales vs. no funcionales), la extracción de información (como entidades de dominio, sus atributos y las relaciones entre ellas), la detección de ambigüedades y defectos lingüísticos, la recuperación de la trazabilidad entre requisitos y otros artefactos de software, y, de manera crucial para esta investigación, la construcción y gestión de glosarios de dominio.

En este contexto tenemos también que una revisión reciente llevada a cabo por Necula et al. (2024) indica que se examinaron más de 130 estudios entre 1991 y 2023 y confirmo que las técnicas de NLP han ido migrando desde modelos cuyo enfoque es reglas estrictas hacia modelos de aprendizaje profundo y LLMs.

En adición recientes investigaciones como la de Firmenich et al. (2025) apuntan que la inclusión de fuentes con una semi estructuración en la presentación de su información aumenta la precisión reduciendo la ambigüedad en la tarea de generación de requerimientos por los sistemas de NLP4RE, aumentando la claridad de estos, y reduciendo los sesgos de los modelos de lenguaje.

Y el aspecto más relevante para este trabajo de investigación es el salto que se produce en 2023, que según una reciente revisión sistemática publicada por Zadenoori et al. (2025) que

análisis 74 estudios primarios publicados entre 2023 y 2024, muestra que los LLMs constituyen ya un nuevo paradigma dentro de la NLP4RE, dado que mientras que para NLP4RE clásico se desarrolló durante décadas (1983-2019) centrándose en tareas analíticas como la detección de defectos, en el campo denominado LLM4RE (Large Language Models for Requirement Engineering) comenzó su recorrido en 2023 y la cantidad de publicaciones relacionadas a este tema crecieron en un 136% en el periodo analizado. Esto demuestra que la adopción masiva de IA gracias a interfaces sencillas como la de ChatGPT, redujeron las barreras técnicas. Este escenario abre una gama de posibilidades para el procesamiento generativo de requerimientos ya que permite vislumbrar un alto potencial para la extracción automatizada de terminologías y la posterior estructuración de glosarios especializados.

2.3 Evolución de la representación vectorial: De TF-IDF a Embeddings.

Para que un algoritmo pueda procesar, analizar o agrupar sobre una fuente de datos de tipo textual, requiere de un proceso de vectorización el cual consta de transformar texto plano a una estructura matemática computable.

En el trabajo de Chang et al. (2021) la técnica de TF-IDF fue utilizada para la minería de documentos de tipo texto, aplicándose sobre resúmenes y palabras clave de artículos científicos. Los autores evaluaban la importancia de las palabras asignándoles un peso según su frecuencia de aparición y empleaban esos valores para su posterior análisis.

Sin embargo y aunque este método es computacionalmente ligero y no requiere de previo entrenamiento con redes neuronales algunas investigaciones subrayan sus deficiencias estructurales cuando se quieren aplicar a escenarios complejos. Como señalan Manning et al. (2008), TF-IDF trata las palabras como si fueran una especie de bolsa de palabras (bag of words) en el que conserva la cantidad de veces que aparece cada término, pero ignora el orden y el contexto en que aparecen. Esto ocasiona que un documento como “el comprador notificó al proveedor” resulte idéntico a “el proveedor notificó al comprador”, puesto que ambos tienen las mismas palabras. Además, mencionan que cada término se trata como una entidad separada e independiente, sin el contexto que le brindan las palabras que la rodean. Y a esto se le suma que las representaciones resultantes son vectores de alta dimensionalidad, tantas como términos haya en el vocabulario, lo que resulta en un costoso cálculo de similitudes entre los documentos.

2.3.1 Word Embeddings estáticos: Word2Vec y GloVe

La primera generación de embedding neuronales fue liderada por Word2Vec (Mikolov et al., 2013) y GloVe (Pennington et al., 2014) estas tecnologías representaron un gran avance. La diferencia con TF-IDF es que estos modelos producen vectores densos de baja dimensionalidad, esto hace posible la captura de relaciones semánticas. Sin embargo, a pesar de estas bondades los modelos presentan una limitación cuando se trata de dominios técnicos como los que podrían encontrarse en IR, esta es que necesitan asignar un único vector por palabra, independientemente del contexto. Esto lo que provoca es que sea incapaz de manejar la polisemia, que es la propiedad de las palabras de tener 2 o más significados dependiendo del contexto en el cual sea usada. Cosa muy presente en glosarios LEL, donde una palabra podría incluso actuar como sujeto, objeto o estado según el símbolo al que pertenezca.

2.3.2 Embeddings contextuales: BERT y Sentence-BERT

El modelo BERT (Bidirectional Encoder Representations from Transformers), fue desarrollado por Devlin et al. (2019), marcó un antes y un después al incorporar representaciones contextuales: es decir que cada token recibe un vector dependiendo de las palabras que la rodeen así resolviendo el problema de la polisemia.

Sin embargo, este método presenta un gran problema a la hora de querer implementarlo para el clustering, este método compara n oraciones a razón de $O(n^2)$. Esto se traduce en que para un glosario LEL con 10000 términos, este método debería hacer 50 millones de operaciones siguiendo la fórmula $\frac{n \times (n-1)}{2}$, haciéndolo inviable. Para resolver este problema Reimers y Gurevych (2019) propusieron Sentence-BERT (SBERT), esta variante del anterior método propone usar redes siamesas para producir embeddings de oración de tamaño fijo comparables. Pasándolo a términos más tangibles la operación antes mencionada, de 10000 términos, con BERT tardaría aproximadamente 65 horas con una gráfica V100, mientras que si se usa la variante SBERT el tiempo cae a 5 segundos aproximadamente.

2.3.3 Embeddings generados por LLM modernos

En un estudio de Tracing the Evolution of Word Embedding Techniques in Natural Language Processing publicado por Nguyen (2026) realiza una comparación entre la era pre GPT-3 y post GPT-3, mediante siete pruebas y reporta un cambio de paradigma con los métodos contextuales y de nivel de oración/documento estos pasan a dominar con un 6.4 veces más odds que en la era de pre GPT.

En el trabajo publicado por Miller y Alexander (2025) muestran que un enfoque basado en embeddings generados por un LLM y agrupados con la técnica Gaussian Mixture Models produce clústeres de texto corto, más interpretables para las personas en comparación con técnicas tradicionales y ampliamente usadas como doc2vec o LDA.

2.4 Aplicación de NLP a la gestión de glosarios LEL

A partir de aquí nos adentraremos en los temas más directamente relacionados con este trabajo de investigación: el procesamiento automático de glosarios de dominio en particular los de tipo LEL.

2.4.1 Extracción de conocimiento de los glosarios LEL a través de PLN

En el trabajo de Antonelli et al. (2024) sus autores proponen y evalúan una forma específica de extracción de conocimiento a partir de un glosario LEL usando técnicas de PLN. En el enfoque usado se utilizan reglas sobre las “kernel sentences” del glosario y también presenta un prototipo funcional, evaluado mediante una escala llamada System Usability Scale, por sus siglas (SUS), obteniendo resultados muy prometedores. En el trabajo también se menciona que el LEL ha sido utilizado en dominios complejos y que su construcción es un proceso iterativo e incremental, en el cual nuevos símbolos pueden emerger de los ya definidos. De aquí se puede deducir entonces que a medida que el glosario se vuelva más detallado y aumente la cantidad de símbolos, este puede volverse más difícil de analizar. A esto se suman aspectos como el relevado por trabajos como el de Roldán Valdiviezo (2025) el cual señala que, si bien el LEL constituye un elemento fundamental para el desarrollo de software, estos tienden a tener estilos de redacción que varían dependiendo de la persona que lo construya, lo que puede incurrir en dificultades para los involucrados a la hora de comprender el documento. Su propuesta para solventar estos inconvenientes formalizando el LEL sintácticamente usando una gramática EBNF (*Extended Backus-Naur Form*). Este aporte evidencia la necesidad de herramientas que ayuden a organizar y estructurar la información para su exitosa posterior interpretación.

2.4.2 LLMs para la construcción y organización de dominio

A la hora de querer organizar términos de un glosario por subdominios surgen algunos problemas relacionados con la identificación de estructuras conceptuales y agrupamientos semánticos coherentes. Este es un problema que guarda estrecha relación con el campo de aprendizaje de ontologías, tal como se documenta en el trabajo de Li et al. (2025) “Large

Language Models for Ontology Engineering”. Este es una revisión sistemática donde analiza y documenta varios trabajos sobre el uso de LLMs en ingeniería de ontologías, producida entre los años de 2018 y 2024, en la que remarca una variedad de estrategias y flujos de trabajo, destacando también algunos desafíos de estandarización.

En la misma línea la revisión del desafío de LLMs4OL (LLMs for Ontology Learning) realizado por Babaei Giglou et al. (2025) se reportó que los sistemas con mejor desempeño en tareas que requieren la estructuración y agrupamiento semántico de términos eran las que combinaban LLMs comerciales con embeddings ajustados al dominio en cuestión. Esto sugiere que combinar representaciones semánticas (embeddings) con instrucciones a LLMs pueden mejorar el desempeño frente a escenarios donde se utilizan estas técnicas en solitario.

2.4.3 LLMs para el agrupamiento semántico de términos

En el ámbito de las IR el estudio de modelos generativos para el apoyo de tareas analíticas intensivas en el lenguaje natural ha recibido una gran atención de parte de los investigadores. Entre ellas la revisión sistemática que realizaron Cheng et al. (2025) en *Software: Practice and Experience* evidenció que las técnicas GenAI son particularmente eficaces cuando hablamos de actividades de análisis que involucran minería de texto, clasificación y agrupamiento de grandes volúmenes de datos. También destacan el uso predominante de enfoques zero-shot y few-shot, los cuales son enfoques que lo que intentan hacer es una actividad exploratoria de análisis ejecutando tareas a partir de instrucciones de lenguaje natural con ningún entrenamiento adicional ni datasets etiquetados, sin utilizar ejemplos en el caso de zero-shot, y muy pocos ejemplos provistos en el prompt para el few-shot. Esta elección está motivada por la escasez de conjuntos de datos etiquetados, situación que también se presenta en los glosarios LEL.

Una revisión publicada por Hemmat et al. (2025) identificaron varias limitaciones en el uso de LLMs en IR, entre estas se encuentran las famosas alucinaciones, validaciones insuficientes a las salidas producidas, algunas dificultades para manejar dependencias complejas y la falta de consistencia en dominios y la necesidad de la aplicación de técnicas de prompt tuning y fine-tuning para tareas específicas de dominio. Además de esto enfatiza en la necesidad de evaluaciones de expertos para corroborar los resultados producidos por LLMs en contextos de IR.

2.5 Clustering aplicado a Ingeniería de Requerimientos

En esta anteúltima sección del estado del arte nos adentramos a los avances realizados por investigadores enfocados en técnicas de clusterización con fuentes de texto en lenguaje natural, sean casos de uso o historias de usuario. Esto es lo más cercano que existe actualmente a la aproximación tomada en este trabajo de investigación.

2.5.1 Clustering sobre casos de uso

Cuando hablamos de clustering en el ámbito de la ingeniería de requerimientos, uno de los trabajos que cobra mayor importancia es el de Kumar et al. (2022) el cual trata de una propuesta de clusterización de historias de usuario mediante el algoritmo del K-Means. Este plantea 4 pasos que van desde tokenización, remoción de stopwords, stemming y lematización. Utilizando como se mencionó el algoritmo K-Means con la validación del coeficiente de silueta. Los autores trabajaron con un total de 98 historias de usuarios, generando de 2 a 14 clústeres. Con esto compararon dos formas de vectorización una usando una tecnología llamada Count Vectorizer y la otra utilizando TF-IDF, y concluyeron que esta última tenía mejores resultados produciendo clústeres más balanceados en especial cuando a medida que aumentaba el valor de k , los resultados eran aún mejores en cuanto a la calidad de los clústeres generados. Sin embargo, debemos recordar que esta tecnología utilizada presenta una gran limitación en lo que a este trabajo de investigación respecta que es la imposibilidad de capturar relaciones semánticas entre palabras, constituyendo así un problema a la hora de lidiar con artefactos con un alto grado de sinonimia y polisemia como lo tienen los glosarios LEL.

En el marco de los glosarios LEL tenemos el trabajo de Delle Ville et al. (2024) en el cual se presenta un enfoque para analizar un conjunto de escenarios por sus similitudes. En el trabajo se comparan escenarios de a pares comparando atributo por atributo: título, objetivo, contexto, actores, recursos y episodios. Como en el trabajo antes mencionado estos atributos pasan por un proceso de lematización y eliminación de las stopwords. Después se analiza si estos son una sola palabra como podrían ser los actores entonces se procesan directamente mientras que si son varias como podrían ser los objetivos entonces se extraen los sustantivos y los verbos. La similitud entre cada par de estos atributos se obtiene mediante un índice Jaccard, que los autores lo definen como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos resultantes donde cuanto mayor es el valor, más similares son los conjuntos, o explicado de forma sencilla es la cantidad de palabras iguales de los atributos comparados dividido el total de palabras con el que consta cada atributo. Los autores mencionan que usaron

cuatro escenarios del ámbito agrícola en donde destacan que obtuvieron buenos resultados donde por ejemplo dos escenarios que describen la misma actividad de riego (“Irrigate by hand” e “Irrigate manually”) alcanzaron un ranking alto de 0,635 puntos, evidenciando su similitud. Mientras que para casos como el riego vs la siembra (“Irrigate with pump” y “Sow tomato seeds”) obtuvieron un ranking de 0,010. Por último, cabe mencionar que los autores de este trabajo enfatizan en que este método tiene un enfoque puramente sintáctico. Esta limitación queda evidenciada por el ejemplo dado por los propios autores, donde mencionan que los sinónimos son considerados como palabras distintas y que palabras como “provide H2O” o “supply water” son consideradas diferentes a pesar de que en el ejemplo dado ambos refieren a la misma acción.

En esta misma dirección Pérez et al. (2025) publicaron una evaluación de modelos PLN preentrenados para medir la similitud de escenarios, pero escritos en español. Los autores mencionan la importancia del trabajo que realizaron mediante la justificación de que la mayoría de los modelos que son preentrenados están desarrollados para el inglés. En el trabajo se seleccionan 14 escenarios de 150 creados por profesionales, al igual que el trabajo anterior todos relacionados con el ámbito agrícola. Y se evaluaron 3 técnicas: TF-IDF, fastText (con un modelo preentrenado en español) y cuatro modelos de SBERT multilingües. Para validar los resultados obtenidos realizaron una serie de encuestas en las que los profesionales indicaban qué escenarios consideraban similares, y los resultados mostraron diferencias significativas. En el trabajo se resaltan las limitaciones de TF-IDF frente a otras aproximaciones, en ejemplos como “Distribuir agua en cultivos” TF-IDF no encontró ningún resultado similar, mientras que los modelos SBERT identificaron correctamente como similares escenarios como “Regar las plantas de tomate” o “Regar las plántulas de tomate” a pesar de que no comparten ninguna palabra con el título buscado. Otro ejemplo que mencionan es el de “Erradicar vegetación indeseada” donde de nuevo TF-IDF no produjo ningún resultado mientras que los modelos de SBERT encontraron similitudes con “Eliminar las malezas” y “Quitar malas hierbas”. Este trabajo demuestra unívocamente la superioridad de los embeddings contextuales frente a las técnicas estadísticas para capturar similitudes semánticas en textos en español.

2.6 Síntesis y brecha identificada

A lo largo de este estado del arte se ha relevado documentación pertinente que va desde el panorama más amplio y general del procesamiento del lenguaje natural aplicado a la ingeniería de requerimientos hasta lo más cercano a este trabajo de investigación. De los cuales se puede concluir que existe una brecha clara que este trabajo intenta abordar.

En primer lugar, se reconoce que el uso del lenguaje natural introduce defectos como la ambigüedad, la vaguedad o la inconsistencia. En este contexto se crean artefactos como los glosarios LEL diseñados para capturar el lenguaje del dominio, lo cual contribuye a reducir estas dificultades, introduciendo el factor de documento semiestructurado con reglas claras de cómo debe ser construido, sin embargo, en dominios que son muy grandes estos documentos pueden llegar a crecer de forma significativa, dificultando así su lectura y navegación por los profesionales.

Con este escenario llegamos a la introducción de técnicas de extracción automáticas para ayudar a los profesionales a lidiar con la extracción de información relevante de la documentación y aumentar su productividad. Frente a esto se consolida el campo de las NLP4RE, que fue estudiado y desarrollado por décadas y que con sus limitaciones facilitan dicha tarea de extracción, los cuales posteriormente fueron evolucionando a las LLM4RE. Con la introducción de los LLMs, las posibilidades se ampliaron considerablemente permitiendo tareas de análisis que antes eran muy difíciles o resultaban directamente inviables para las otras técnicas.

También se evidenciaron las limitaciones de enfoques como el de TF-IDF el cual ha demostrado su incapacidad para la comprensión de sinónimos o relaciones contextuales. En contraste con otras técnicas como son los modelos contextuales SBERT, los cuales han demostrado ser idóneos en tareas donde se debe entender el contexto de las palabras analizadas.

Ya hablando del área específica de la clusterización algunos trabajos han aplicado dichas técnicas en elementos relacionados con la ingeniería de requerimientos como, por ejemplo, historias de usuario o escenarios. Pero algunos de estos enfoques relevados presentan limitaciones como operar a nivel puramente sintáctico, o que dependan de representaciones estadísticas que no capturan la semántica del dominio. También se abordaron aproximaciones multilingües, específicamente español, y se obtuvieron resultados muy prometedores de comprensión de escenarios utilizando las herramientas de SBERT, incluso demostrando la idoneidad de los modelos contextuales al solucionar el problema existente en los anteriores trabajos relacionados a la correcta interpretación semántica.

A partir del análisis realizado entonces se identifica que ninguno de los trabajos relevados aplica técnicas de clustering en documentos de tipo glosarios LEL los cuales como ya se ha mencionado tienen una estructura y diseño particular. Por lo tanto, la brecha de investigación se encuentra en el diseño de un enfoque que mediante el uso de técnicas modernas de representación semántica (embedding) pueda aplicar una evaluación comparativa entre múltiples algoritmos de clustering (K-Means, DBSCAN, agrupamiento jerárquico

aglomerativo, Gaussian Mixture Model y Affinity Propagation) y un enfoque generativo basado en LLMs, sobre este tipo específico de documentación. Esto con el fin de determinar cuáles resultan más efectivos a la hora de la identificación automática de subdominios. Este enfoque permitirá superar las limitaciones de los modelos sintácticos y contribuir a una comprensión más estructurada y navegable del conocimiento del dominio.

Capítulo 3 - Marco teórico

3.1 Ingeniería de Requerimientos

La ingeniería de requerimientos cumple un papel fundamental en el proceso de producción de software puesto que esta se centra en la elicitación de los requerimientos o, dicho en otras palabras, define lo que se quiere producir. Su principal tarea consiste en generar las especificaciones correctas que describan con claridad, sin ambigüedades y en forma consistente y compacta las necesidades de los clientes.

Habiendo establecido la importancia de la ingeniería de requerimientos en el desarrollo de software, debemos atender algo antes de pasar a tratar los aspectos referentes a la administración adecuada de los requerimientos, es importante primero definir lo que es un requerimiento y cuáles son las características que estos tienen.

3.1.1 Requerimientos

Para el Institute of Electrical and Electronics Engineers (IEEE, 1990) el significado de un requerimiento en el ámbito de sistemas es una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

También Sommerville (2005) menciona que un requerimiento es una descripción de un servicio que un sistema debe proporcionar o también menciona que podría ser una limitación de este.

Teniendo en cuenta ambas definiciones antes mencionadas podemos decir que un requerimiento es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso.

3.1.2 Tipos de requerimientos

Los requerimientos pueden dividirse en 3 grupos: requerimientos funcionales, los no funcionales y los de dominio.

Para Sommerville (2005) los requisitos funcionales son los servicios que debe proporcionar el sistema y cómo debe comportarse ante determinadas entradas o situaciones. Mientras que los no funcionales menciona que son restricciones o limitaciones que presenta el sistema, incluyéndose también cosas como las limitaciones en el tiempo de desarrollo del sistema o estándares que debe contemplar el mismo. Mientras que los de dominio son aquellos

que provienen del dominio de aplicación del sistema, los cuales reflejan características y restricciones de este.

En otras palabras, los requerimientos funcionales son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Mientras que los no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como, por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, etc. Y los requisitos de dominio serían más enfocados al rubro para el que se realiza el software, como por ejemplo en el ámbito bancario un requisito de dominio sería que el sistema debe cumplir con las normativas de antilavado de dinero.

3.1.3 Importancia de ingeniería de requerimientos (IR)

El término de ingeniería de requerimientos se utiliza para englobar todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto de software determinado. Según Sommerville (2005). La IR es el proceso de comprender y definir qué servicios se requieren en el sistema, identificando además sus restricciones sobre el desarrollo y funcionamiento de este. Esta es una etapa particularmente crítica, debido a que los errores que se cometan en ella se transformarán en problemas posteriores de implementación. El autor señala que los costes de corregir uno de estos problemas, que sean descubiertos una vez en desarrollo o con el sistema funcionando, son mucho mayores que el de reparar errores de diseño o codificación, porque obliga a modificar otra vez el diseño y la implementación y volver a probar todo el sistema. Para poder gestionar esta etapa del desarrollo, se organiza en cuatro fases: el estudio de viabilidad, obtención y análisis, especificación y por último validación. El resultado de estas cuatro etapas es el documento de requerimientos que actúa como la declaración acordada de lo que el sistema debe hacer, este además sirve para la planificación del desarrollo y el mantenimiento del sistema. Además, esta validación comprueba la validez, la consistencia, la completitud, el realismo y la verificabilidad de los requerimientos apoyándose en el hecho de que los clientes lean ese documento para poder confirmar que satisface sus necesidades. De este modo, el documento contribuye a la calidad del producto, reduce el riesgo de que el sistema no cumpla las expectativas de los clientes y convierte al documento en una forma de comunicación entre los desarrolladores y los clientes.

3.1.4 Glosarios de dominio

Los glosarios de dominio para la IR son documentos con una estructura en particular cuyo objetivo es brindar la información necesaria de la forma más clara posible a los analistas que trabajan en el desarrollo de un software en particular. Según Arora et al. (2017) un glosario es una parte muy importante de un documento de requerimientos de software, debido a que su objetivo es intentar presentar términos técnicos y definiciones de un dominio de forma explícita. Lo que ayuda a reducir las imprecisiones y la ambigüedad.

3.2 Léxico Extendido del Lenguaje (LEL)

El léxico extendido del lenguaje o también conocido por sus siglas LEL, actúa como un glosario que intenta describir el lenguaje de un dominio de aplicación sin centrarse en la aplicación en sí. Según sus creadores Leite y Franco (1993) en su trabajo “*A strategy for conceptual model acquisition*” definen a LEL como un glosario que registra y describe el vocabulario de una aplicación, permitiendo comprender los conceptos del dominio sin requerir un conocimiento profundo de la funcionalidad del proceso de negocio.

Este organiza el lenguaje mediante símbolos, los cuales representan términos o expresiones específicas del universo que se está analizando. Estos símbolos incluyen un nombre (o varios para cubrir los sinónimos) para identificación, y según Antonelli et al. (2024), se describe mediante dos atributos: la noción (significado literal o técnico del símbolo) y la respuesta de comportamiento (las acciones, condiciones o relaciones asociadas al símbolo en el contexto del dominio). Estos son explicados mediante oraciones que deben seguir dos principios según Leite y Franco (1993), el primero es el principio de circularidad el cual establece que se debe hacer un uso intensivo de los símbolos ya definidos para nuevas definiciones. El segundo es el principio de vocabulario mínimo, este establece que se debe limitar el uso de términos que sean ajenos al LEL en descripciones de los símbolos.

3.2.1 Estructura de los Glosarios LEL

Los glosarios LEL, según Antonelli et al. (2024), organizan cada uno de los símbolos pertenecientes a un dominio de aplicación en cuatro categorías fundamentales:

- **Sujetos:** Son las entidades que actúan o realizan tareas. Se describen por sus características (noción) y por las acciones que llevan a cabo (respuesta de comportamiento).

- **Objetos:** Son las entidades que reciben las acciones de los sujetos. Su noción se refiere a ¿qué es el objeto?, y su respuesta de comportamiento a las acciones que les afectan.
- **Verbos:** Representan las acciones que los sujetos realizan sobre los objetos. La noción explica el propósito de la acción, y la respuesta de comportamiento detalla los pasos necesarios para ejecutarla.
- **Estados:** Indican las condiciones o situaciones en las que se encuentran los sujetos u objetos. La noción describe esa situación, y la respuesta de comportamiento señala cómo se puede verificar esa situación.

Según Leite y Franco (1993) cada símbolo en el LEL debe tener un nombre del dominio que debe ser representativo, agrupando sinónimos bajo un mismo término y capturando aquellos que parezcan tener un significado especial dentro del dominio de la aplicación. Es importante que cada símbolo se relacione con otros, manteniendo una estructura interconectada y coherente.

Ejemplo práctico real

A continuación, mostraremos un ejemplo real de cada tipo de símbolo (sujeto, objeto, verbo y estado) el glosario LEL se utiliza para realizar las pruebas de este trabajo de investigación.

Tabla 2*Descripción de un sujeto del glosario LEL*

ATRIBUTOS	DESCRIPCIÓN
Sujeto	Tutor
Noción	Persona mayor de 18 años que se encarga de manejar el sistema para los pacientes menores de 18 que tutela. Se registran los siguientes datos: nombre, apellido, teléfono, mail, dirección. Ingresa al sistema con el dni del paciente tutelado. Un tutor también puede estar registrado en el sistema como paciente.
Impactos	• El tutor registra a un menor en el sistema. • El tutor ingresa al sistema.

Nota. Elaboración propia.

Tabla 3*Descripción de un objeto del glosario LEL*

ATRIBUTOS	DESCRIPCIÓN
Objeto	Consentimiento informado
Noción	Documento que el paciente o tutor debe firmar para aprobar la realización del estudio. Contiene el consentimiento informado (texto, hardcodeado en la base de datos del laboratorio). Hay un consentimiento informado para cada Tipo de Estudio. Cuando el paciente firma el consentimiento informado pasa al estado firmado.
Impactos	• El empleado baja el consentimiento informado. • El empleado carga el consentimiento informado firmado. • El sistema envía el consentimiento informado por mail al paciente.

Nota. Elaboración propia.

Tabla 4*Descripción de un verbo del glosario LEL*

ATRIBUTOS	DESCRIPCIÓN
Verbo	Ver paciente
Noción	Acción por la cual un empleado ve los datos de un paciente
Impactos	• El empleado identifica un paciente. • El sistema devuelve los datos del paciente.

*Nota. Elaboración propia.***Tabla 5***Descripción de un estado del glosario LEL*

ATRIBUTOS	DESCRIPCIÓN
Estado	Estudio anulado por falta de pago
Noción	El estado al que llega un estudio cuando pasaron 30 días de cargado el comprobante de pago y no se pagó
Impactos	• El empleado visualiza los estudios anulados por falta de pago.

Nota. Elaboración propia.

3.3 Inteligencia Artificial y Procesamiento de Lenguaje Natural

3.3.1 Fundamentos del PLN

El procesamiento del lenguaje natural, o “PLN” por sus siglas, es un subcampo dentro de la informática y de la inteligencia artificial que utiliza el aprendizaje de máquina (machine learning) para hacer que las computadoras entiendan, generen y respondan al lenguaje humano (Holdsworth y Stryker, s. f.).

Esto se logra basándose en tres pilares importantes: la lingüística, la informática y la matemática, según Jurafsky y Martin (2008). Los autores también mencionan que, desde el punto de vista de la lingüística, se investiga la sintaxis, la morfología, la semántica y la pragmática. Desde las matemáticas, se utilizan modelos probabilísticos, modelos basados en álgebra lineal y lógica formal. Y a nivel informático se desarrollan los algoritmos de búsqueda, aprendizaje automático y las implementaciones computacionales.

3.3.2 Tareas fundamentales PLN

El PLN tiene una serie de tareas que debe hacer antes de poder procesar un texto, ya sea aplicando vectorización o clustering, las tareas que aquí se desarrollarán en este apartado son las pertinentes para este trabajo de investigación, no se descarta la existencia de más técnicas que las que aquí se mencionan.

Tokenización

La primera técnica es la tokenización, que según Jurafsky y Martin (2008) es el proceso por el cual un texto es dividido en pequeñas partes, palabras y oraciones, conocidas como tokens. También menciona que, en un idioma como el inglés, lenguaje en el que se tratan estos temas en el libro, las palabras están separadas por espacios, y que a pesar de que esto pueda parecer trivial, él revela que los espacios por sí solos no son suficientes para determinar los límites de las palabras.

Ejemplo de tokenización:

Hoy estoy atravesando un gran dolor, ¿podrías ayudarme a aliviarlo?

Luego de aplicar la tokenización el texto quedaría de la siguiente forma:

[“Hoy”, “estoy”, “atravesando”, “un”, “gran”, “dolor”, “,”, “¿”, “podrías”, “ayudarme”, “a”, “aliviarlo”, “?”]

Lematización

La segunda técnica es la de lematización. En el libro de Jurafsky y Martin (2008), se define un lema como la forma gramatical que se usa para representar una unidad léxica, siendo esta la forma base de una palabra. Dicho en otras palabras, toma una palabra y la lleva a su representante en el diccionario, por ejemplo, los verbos los lleva a su infinitivo, palabras como “corriendo”, “correrán” se convierten en “correr”. Mientras que sustantivos y adjetivos los lleva generalmente al masculino singular (por ejemplo, “gatos” o “gatas” pasan a ser “gato”).

El proceso de convertir una palabra flexionada a su lema, o palabra base, se denomina lematización. Esta tarea resulta fundamental para el análisis de textos de IR porque permite que variaciones morfológicas de un mismo término sean tratadas como una sola unidad.

Remoción de stopwords

La tercera y última que mencionaremos es la remoción de las stopwords. Pero primero debemos saber qué son las stopwords, según Kumar et al. (2022) estas son palabras que carecen de significado relevante para un análisis, tales como preposiciones, pronombres o artículos. La eliminación de estas reduce el ruido en los datos resultantes y permite que técnicas posteriores se centren en las palabras que realmente aportan un valor semántico para el análisis que se quiere llevar a cabo.

Estas tres tareas de preprocesamiento tokenización, lematización y remoción de stopwords conforman la base sobre la cual se aplican las técnicas de representación vectorial y agrupamiento que se describen en las siguientes secciones.

3.3.3 Representación vectorial de texto (*embedding*)

Los algoritmos de aprendizaje automático, como las técnicas de clusterización que se utilizarán en la realización de este trabajo, utilizan datos numéricos, ya que necesitan calcular distancias y promedios entre los elementos que deben analizar. En texto con lenguaje natural esto es muy complicado ya que gracias a su naturaleza simbólica no puede ser procesado directamente por los algoritmos de estas técnicas. Por esto, es necesario transformar el texto de lenguaje natural a representaciones numéricas que posibiliten su posterior procesamiento por dichos algoritmos. Según Jurafsky y Martin (2008), los modelos de espacio vectorial constituyen la base de la recuperación de información como de los tratamientos computacionales del significado de las palabras.

Esta transformación es de vital importancia en cualquier proceso de PLN aplicado a ingeniería de requerimientos. Por ejemplo, Kumar et al. (2022) convirtieron historias de usuario en vectores numéricos mediante TF-IDF y Count Vectorizer antes de aplicar el algoritmo del K-Means, porque este algoritmo necesita representaciones numéricas para calcular centroides y asignar cada elemento al clúster más cercano.

TF-IDF en PLN

El término de TF-IDF viene de su nombre en inglés “*Term Frequency-Inverse Document Frequency*”, según Manning et al. (2008) es un esquema estadístico usado para saber el peso cuantitativo de una palabra dentro de un documento, determinando su relevancia también a través de la colección de documentos a analizar. Los autores mencionan que por un lado la frecuencia del término o TF, por sus siglas en inglés, es la encargada de comprobar cuántas veces aparece una palabra dentro de un mismo documento, bajo la premisa de que mientras más veces aparezca dicha palabra más importante o informativa será con respecto al

documento analizado. Por otro lado, la frecuencia inversa del documento o IDF, por sus siglas en inglés, evalúa la presencia de la palabra a lo largo de los documentos de la colección a analizar. Esto mediante un logaritmo del total de documentos dividido el número de documentos que contienen dicha palabra. Esto bajo el objetivo de penalizar las palabras que sean muy comunes entre los documentos, debido a que estas no otorgan distinción alguna entre los documentos, o dicho de otra forma no aportan valor para diferenciar a un documento de otro.

Modelos de Lenguaje Grandes (LLMs)

Los LLM son modelos de aprendizaje profundo que según Amazon Web Services (s. f.) son capaces de comprender tanto gramática, idiomas e incluso conocimiento general a través del autoaprendizaje. La denominación de “grandes” corresponde a dos factores: el primero porque estos modelos son entrenados con enormes cantidades de datos de texto y código, los cuales abarcan miles de millones de páginas web que provienen de internet como Common Crawl y Wikipedia. Y el segundo factor es la inmensa cantidad de parámetros que componen la red neuronal.

Su arquitectura está basada en una tecnología llamada transformadores, propuesta por Vaswani et al. (2017), en la propuesta original, consta de dos componentes principales: un codificador (encoder) y un decodificador (decoder). El codificador recibe una secuencia de entrada y entonces genera representaciones internas para cada elemento, mientras que el decodificador usa esas representaciones para generar una secuencia de salida. Los autores también mencionan que en los modelos recurrentes previos los textos tenían que procesarse de forma secuencial, haciendo que para entender y procesar la última palabra de una oración debían procesarse todas las anteriores una por una. Con esta nueva arquitectura se propuso que se reemplace esa funcionalidad por completo con otra llamada self-attention, la cual permite al modelo considerar todas las palabras simultáneamente y determinar cuáles son relevantes entre sí, sin importar su posición dentro de la oración. Esta facultad es la que les permite a estos modelos procesar esas grandes cantidades de datos, y también entender las relaciones entre palabras en un contexto muy amplio.

Embeddings contextuales: BERT y SBERT

Entre los modelos de embeddings más importantes para este trabajo de investigación se encuentran BERT y su variante SBERT, el primero de ellos se basa en el modelo Transformer y se preentrena mediante dos tareas no supervisadas. La primera es el modelado de lenguaje enmascarado. El cual consta de ocultar el 15% de los tokens de entrada, y el modelo debe predecirlos utilizando solamente el contexto brindado por los tokens que no fueron ocultados.

La segunda de estas tareas es la de predicción de la siguiente oración, en donde el modelo recibe dos oraciones y debe decidir si estas son contiguas o no, es decir, si la segunda es la continuación de la primera o por el contrario es una oración cualquiera tomada del corpus. El modelo BERT fue preentrenado sobre el corpus BooksCorpus (800 millones de palabras) y la Wikipedia (2500 millones de palabras) y se presenta en dos variantes, una es el BERT-Base el cual consta de 12 capas de Transformer y 110 millones de parámetros. Y la otra es BERT-Large la cual consta de 24 capas y 340 millones de parámetros (Devlin et al., 2019).

El segundo modelo es el SBERT (Sentence-BERT) este es una adaptación del modelo BERT, el cual se utiliza para generar representaciones vectoriales de oraciones de forma que aquellas que sean semánticamente similares queden próximas entre sí en el espacio vectorial. Para lograrlo este parte de un modelo BERT ya entrenado y lo ajusta mediante redes siamesas o tripletas, en la estructura siamesa, dos ramas de un mismo modelo BERT procesan dos oraciones distintas, no son dos modelos distintos sino el mismo en dos ramas. Para lograr obtener una representación vectorial de la oración procesada cada entrada es procesada por un modelo BERT, el cual produce un vector por cada token de la oración. Luego se le realiza una operación de pooling, esta puede realizarse mediante varios métodos alternativos, el artículo propone estrategias como el uso del token CLS, el promedio de los vectores (MEAN) o el máximo, por defecto, utiliza la estrategia MEAN. Que consiste en promediar las representaciones generadas, esto para transformar esos vectores en uno solo que represente a la oración. Sin embargo, esto no es suficiente para obtener buenos resultados. Para que los vectores resultantes capturen las similitudes semánticas correctamente es necesario un paso más, uno llamado “fine-tuning”. Este ajuste consiste en ajustar un modelo BERT pre entrenado, mediante la tarea de inferencia de lenguaje natural (NLI). En esta tarea el modelo recibe oraciones de a pares las cuales debe clasificar según la relación que exista entre ellas, según las categorías de: implicación (cuando una se deduce de la otra), contradicción (cuando una niega la otra) o neutral (cuando no tienen relación directa). Esto se realiza sobre dos datasets de NLI uno es el SNLI, que contiene 570 mil pares de oraciones y el otro es MultiNLI que contiene 430 mil pares. Al terminar este entrenamiento extra, el modelo de SBERT ahora entiende qué hace que dos oraciones sean similares o no, y eso luego se refleja en qué tan lejos o cerca quedan sus representaciones vectoriales (Reimers y Gurevych, 2019).

Una vez obtenidos los vectores resultantes estos pueden compararse mediante medidas como la similitud coseno para determinar su grado de similitud.

Embeddings basados en LLMs

Los embeddings con LLMs han tomado mucha relevancia en los últimos años, estos además de ser usados como sistemas generativos comenzaron a tomar un papel más presente en la producción de representaciones vectoriales. Esto gracias a los avances en modelos como GPT, LLaMA o Mistral. Existen dos estrategias que se pueden usar para derivar embeddings mediante un LLM. El primero es mediante prompts directos, sin realizar el ajuste previo adicional específico para la tarea de embeddings. La segunda, por otro lado, es realizando ese ajuste adicional previo a la solicitud al modelo, el objetivo de este ajuste va orientado a mejorar la calidad de los embeddings generados, esto lo realiza incluyendo aspectos como la arquitectura utilizada, los objetivos de aprendizaje, la construcción de datos y las técnicas de ajuste (Tao et al., 2024).

En el contexto de este trabajo de investigación esta clase de embeddings será bastante valiosa debido a que según el trabajo de Tao et al. (2024) los LLMs como modelos para los embeddings al tener mayor capacidad expresiva, mayor cantidad de parámetros y corpus de entrenamiento más extensos, tienen gran capacidad de capturar relaciones semánticas complejas. Estas características son muy útiles para utilizarlos en técnicas de clusterización.

3.3.4 Técnicas de Prompting

Cuando se habla de prompting se hace referencia al acto de enviarle una instrucción, pregunta o consigna (prompt) a un modelo de lenguaje, para que este formule una respuesta. Dentro de esta práctica existen diferentes técnicas que guían el comportamiento del modelo de lenguaje, y las dos más fundamentales son el zero-shot y el few-shot. Según Winland (s. f.) el zero-shot consiste en darle al modelo una tarea, o pregunta sin ningún ejemplo de lo que se espera como respuesta, confiando meramente en el conocimiento adquirido mediante su entrenamiento para interpretar y responder a ese prompt proporcionado. Por otro lado, few-shot es una técnica que consiste en pasarle al modelo algunos ejemplos dentro de ese prompt proporcionado, para mostrarle al modelo el patrón que debe seguir al momento de responder, con esto el modelo puede analizar los ejemplos y estructurar su respuesta siguiendo esas pautas proporcionadas. La diferencia principal entre ambos es que el zero-shot sirve más para tareas genéricas donde la interpretación estándar es suficiente y permite tener respuestas rápidas, mientras que el few-shot sirve más en escenarios en donde los resultados necesitan ser más consistentes en tareas especializadas que necesitan tener criterios particulares.

Modelo de espacio vectorial y similitud coseno

El modelo de espacio vectorial es una representación matemática en donde se representan documentos como vectores de un espacio común, cada dimensión de este espacio representa un término o característica mientras que sus valores reflejan el peso de esos términos en dicho documento, es decir, qué tan importante es el término. Esta representación otorga la posibilidad de comparar documentos entre sí, al representarlos como un vector de pesos esta comparación se logra mediante unas operaciones matemáticas. La forma más estándar de hacer esta medición es mediante la similitud coseno, esta calcula el coseno del ángulo entre dos vectores, y cuanto menor es dicho ángulo mayor similitud tendrán los documentos. Este método cuenta con la ventaja de que no es afectado por la longitud de los documentos comparados, debido a que se enfoca en la dirección de los vectores que los representan y no en su magnitud (Manning et al., 2008).

Otra forma de realizar esta medición es a través de la distancia euclídea, también conocida como norma L2, esta al igual que la anterior descripta mide qué tan lejos se encuentran dos puntos en un espacio vectorial. Esta medición se logra mediante la longitud de una línea recta que une ambos puntos, calculada a partir de las coordenadas de estos. En el contexto de clustering esta medida se utiliza para cuantificar qué tan similares son dos documentos, bajo la premisa de que cuanto menor sea la distancia entre ellos, mayor parecido tendrán los documentos en cuanto a su contenido (Manning et al., 2008).

Capítulo 4 – Contribución

En este capítulo se inicia realizando una introducción a los clústeres utilizados en este trabajo y a sus aspectos más relevantes para luego presentar los resultados obtenidos por la experimentación llevada a cabo sobre el glosario LEL suministrado. Como se menciona en la metodología, cada símbolo fue tratado como una unidad de análisis independiente, siendo estos representados mediante la concatenación de su nombre, su noción y su impacto puesto que estos campos tienen la información necesaria para crear los subdominios.

El análisis se organiza en torno a dos preguntas de investigación planteadas, determinar si las técnicas de clusterización permiten identificar subdominios y en caso afirmativo establecer qué combinación de representación vectorial y algoritmo obtiene mejores resultados, usando las métricas externas e internas.

4.1 Clustering

El clustering es una técnica que se caracteriza por ser no supervisado. Este organiza y clasifica diferentes objetos o puntos de datos en grupos denominados clústeres basándose en criterios de similitud, de modo que los elementos que pertenecen a un mismo grupo sean más similares entre sí, que respecto de los elementos de los que se componen otros grupos (Jain et al., 1999).

Que el clustering sea una técnica no supervisada, para Jain et al. (1999) implica que el algoritmo debe trabajar con un conjunto de datos que no ha sido etiquetado con resultados predefinidos. En lugar de ser supervisado por datos que ya hayan sido clasificados y etiquetados previamente, el algoritmo debe encontrar por sí mismo las estructuras, patrones y relaciones en los datos.

En el clustering un aspecto que resulta central es la elección de medida de similitud o distancia, debido a que distintas formas de medir proximidad pueden resultar en agrupamientos distintos. Para nuestro caso, en donde debemos procesar documentos textuales, estos no pueden ser agrupados directamente en su forma original, sino que primero debemos representarlos mediante algún tipo de valor numérico (Manning et al., 2008), como los que produce TF-IDF, embeddings contextuales o los embeddings generados por LLMs. En base a esta representación el algoritmo del clustering ya puede realizar cálculos en base a las distancias y analizar qué elementos (entradas, términos o definiciones) se encuentran más próximos que otros y asignarlos a un mismo grupo.

En términos más generales, el clustering puede entenderse como una técnica que nos sirve para explorar un conjunto de datos, esta es particularmente útil cuando se trata de datos de los que no se conoce previamente sus categorías internas. Jain et al. (1999) indica que esta técnica ha sido usada en múltiples contextos o disciplinas debido a su utilidad como herramienta de análisis exploratorio.

4.1.1 Tipos de Clustering

Cuando hablamos de tipos de clustering IBM (s. f.) menciona que no existe solo una forma de realizar clustering, ya que los grupos pueden definirse a través de diferentes criterios. Las características del conjunto de datos son el factor decisivo al momento de elegir el algoritmo. Entre estas características se encuentran: el tamaño de los datos ingresados, que tan estrictas o flexibles sean las fronteras entre los clústeres que se formarán, cuántas características o atributos tienen cada elemento del glosario y el número de clústeres que posea este conjunto de datos. Ya que un algoritmo puede resultar adecuado para ciertos datos, pero no necesariamente ofrecer buenos resultados en otro contexto.

4.1.2 Técnicas de clustering

Existen múltiples enfoques desde los que se pueden aplicar la clusterización. Para mantener la simplicidad de la investigación solo detallaremos algunos de los más utilizados y que sean relevantes para la misma.

Clustering basado en centroides

De acuerdo con IBM (s. f.) este tipo de clustering es aquel que “divide un conjunto de datos en grupos similares en función de la distancia entre sus centroides” (sección “Clustering basado en centroides”). Siendo el centroide de cada clúster la media de los elementos, representados por puntos que pertenecen a ese clúster.

La técnica más famosa que está basada en este enfoque es la de K-Means. Según el trabajo de Jain et al. (1999) este es un algoritmo de particionamiento que se enmarca en los métodos basados en el criterio de error cuadrado. En términos tangibles, el algoritmo busca lograr que los centroides de los clústeres resultantes sean lo más representativo posible de los elementos que componen dicho clúster.

Su funcionamiento se basa en cuatro etapas en un proceso iterativo, la primera de ellas la etapa de selección de los centros iniciales de los clústeres, denominados centroides, el algoritmo debe seleccionar la cantidad (K) de centroides iniciales ya sea por K patrones elegidos aleatoriamente del conjunto de datos o con K puntos definidos al azar dentro del

espacio que contiene a los datos. En la segunda etapa tenemos la etapa de asignación, en la cual el algoritmo calcula la distancia entre cada elemento del conjunto de datos con respecto a los centroides existentes, asignándolo al clúster cuyo centroide tenga mayor similitud, que esté más cercano. La tercera etapa es la del recálculo, en donde se debe recalcular la posición de los centroides. Esta nueva posición se obtiene calculando el promedio de las posiciones de los elementos que fueron asignados a ese clúster en la iteración actual. Y como última etapa el algoritmo debe evaluar si se llegó al criterio de convergencia. Este criterio se alcanza cuando los elementos asignados a los clústeres no sufren reasignaciones a otros clústeres, si estos se reasignan entonces se vuelve a la etapa dos (asignación). Esta iteración se detiene ya sea cuando se alcanza ese criterio de convergencia o cuando las mejoras obtenidas de una iteración a la otra resultan despreciables (Jain et al., 1999).

Cabe destacar que, según Jain et al. (1999) este algoritmo presenta algunas contras. La primera de ellas es la necesidad de definir de antemano la cantidad de clústeres (K), esto resulta particularmente problemático cuando se trata de fuentes de datos en donde no contamos con la información de cuántas agrupaciones naturales existan en los datos, como es el caso de los glosarios LEL. El segundo problema es que el algoritmo tiende a generar clústeres con una forma circular y si la geometría de los agrupamientos en los clústeres no resulta ser esa, el desempeño del algoritmo puede verse comprometido. Como última contra remarcada por el autor, tenemos que la elección aleatoria de los centroides al inicio del proceso iterativo puede resultar desfavorable para el resultado final. Esto es porque la posición inicial de los centroides condiciona cómo se formarán los primeros grupos y esto afecta las iteraciones siguientes. K-Means no evalúa todas las posibles formas de agrupar los datos, sino que parte de una posición aleatoria inicial, y si esta es desfavorable, entonces el algoritmo puede llegar a un resultado final que, si bien no es mejorable mediante reasignaciones, podría no ser la mejor solución que se pueda obtener. A este último inconveniente se lo conoce como convergencia a un mínimo local.

Clustering jerárquico

Este tipo de clusterización, para IBM (s. f.) se trata de uno “basado en la conectividad, agrupa los puntos de datos en función de la proximidad y la conectividad de sus atributos” (sección “Clustering jerárquico”). Este tipo de clusterización determina los clústeres por aproximación de los elementos a procesar, siendo los más próximos los que guardan mayor relación y los más alejados los que menos, en base a esto se crean los clústeres. Cabe destacar que este método a diferencia de K-Means no necesita que se especifique el número de clústeres

previo al proceso de datos, sino que este crea una red jerárquica de grafos de los clústeres en cada nivel, pudiéndose representar mediante un dendrograma.

Uno de los enfoques para realizar clustering jerárquico, y el más relevante para nuestra investigación, es el aglomerativo. Según Jain et al. (1999), comienza considerando a cada elemento como un clúster individual pero luego los va fusionando poco a poco. El resultado de aplicar este enfoque es un dendrograma, el cual es una estructura de árbol jerárquico que representa las fusiones de clústeres que se van realizando y el nivel de similitud en los que dichas fusiones ocurrieron. Este árbol puede cortarse en cualquier nivel que se desee y dependiendo la altura a la que se corte se obtendrán distintas cantidades de clústeres. Esta es una característica valiosa debido a que le permite a la persona que ejecuta este enfoque, explorar los datos en distintas granularidades sin tener que ejecutar el algoritmo varias veces.

El funcionamiento de este algoritmo puede dividirse en tres pasos iterativos según Jain et al. (1999). El primero de ellos es calcular una matriz de proximidad, la cual contendrá la distancia entre cada par de elementos, y se trata a estos elementos como clústeres individuales. El segundo paso se trata de identificar los elementos (clústeres) más cercanos en proximidad, para esto se usan los datos que se almacenan en la matriz ya mencionada. Y como último paso es fusionar esos clústeres, dejando representada dicha fusión en la matriz de proximidad, y luego volver a hacer el cálculo de la distancia existente entre este nuevo clúster y los restantes. Este proceso se repite hasta que todos los elementos quedan en un único clúster, de esta forma la salida del algoritmo es un dendrograma con la secuencia de todas las fusiones.

Una de las limitaciones que se presentan al usar este algoritmo, es su costo a nivel computacional. Debido a que al requerir almacenar en una matriz de tamaño “ $n \times n$ ”, donde n es la cantidad de elementos a agrupar, las distancias o similitudes entre pares de elementos o clústeres, esta debe mantenerse en memoria a lo largo de todo el proceso, y dependiendo del volumen de datos con los que se requiera trabajar, esto puede resultar muy costoso (Jain et al., 1999).

Clustering basado en densidad.

Para IBM (s. f.) el clustering basado en densidad “funciona mediante la detección de áreas donde se concentran puntos y donde están separados por áreas que están vacías o escasas” (sección “Clustering basado en densidad”), es decir que este método de clustering identifica clústeres a partir de regiones donde los datos se concentran mucho, y los separa de otros mediante zonas de baja densidad de datos. Este tipo de clustering es muy útil cuando se trabaja con fuentes de datos con “ruido” o valores atípicos ya que tiene la capacidad de distinguir entre datos que se consideran ruido y los que pertenecen a un clúster.

El algoritmo que representa este enfoque es uno llamado *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) desarrollado por Ester et al. (1996) quienes mencionan que el algoritmo opera con dos parámetros: ϵ (épsilon) el cual define un radio alrededor de cada elemento del conjunto de datos, y busca cuántos otros elementos se encuentran dentro de ese radio, el otro parámetro es MinPts, que establece la cantidad mínima de elementos que deben encontrarse dentro de ese radio para considerar que esa zona posee densidad suficiente como para formar parte de un clúster. En base a estos parámetros DBSCAN clasifica a un elemento en alguna de tres categorías:

- Core Point: Un elemento es un Core Point si dentro de su radio ϵ , se encuentran al menos MinPts cantidad de elementos, lo cual indica que se encuentra en una zona de alta densidad.
- Border Point: Un elemento entra en esta categoría si no puede entrar en la categoría anterior, pero se encuentra dentro del radio ϵ , de algún Core Point. Dicho de otra forma, se encuentra en el borde de un área de alta densidad.
- Noise Point: Esta categoría es para los elementos que no encajan en ninguna de las categorías anteriores. Lo cual es indicio de que el elemento se encuentra en un área de baja densidad sin conexión con ningún agrupamiento.

Ester et al. (1996) describen el funcionamiento de este algoritmo de la siguiente manera, DBSCAN selecciona un elemento del conjunto que aún no ha sido procesado. Si al examinar los elementos vecinos que tenga dentro de su radio ϵ , se encuentra con MinPts cantidad de estos, entonces lo cataloga como Core Point y a partir de él inicia el nuevo clúster. Luego de esto el algoritmo pasa a examinar cada uno de los elementos que se encuentran dentro del radio mencionado y los incorpora a ese clúster iniciado, si alguno de estos elementos resulta ser también un Core Point entonces examina los elementos vecinos de este también, siempre aquellos que estén dentro del radio ϵ , y los elementos encontrados se incorporan al nuevo clúster, este proceso continúa hasta que ya no se encuentren Core Points que aporten elementos nuevos al clúster. Si por el contrario el elemento inicial analizado no es un Core Point, se categoriza temporalmente como ruido, esta categorización es temporal debido a que puede ser que este elemento esté dentro del radio ϵ de algún Core Point descubierto más adelante.

DBSCAN presenta algunas limitaciones que deben ser tenidas en cuenta. La primera es la necesidad de establecer valores para los parámetros ϵ y MinPts, parámetros que se aplican a todos los clústeres del conjunto de datos. Esto significa que la elección errónea de estos valores puede significar que, si dentro de un mismo conjunto de datos coexisten clústeres más densos

que otros, los valores elegidos podrían no servir para detectar los menos densos o viceversa o podría llegar a fusionar dos clústeres cercanos, que no corresponda (Ester et al., 1996).

Affinity Propagation

El algoritmo de Affinity Propagation según sus creadores Frey y Dueck (2007) tiene una diferencia en cuanto al funcionamiento respecto a algoritmos como K-Means o el de clustering jerárquico aglomerativo, esta radica en que Affinity Propagation busca identificar los elementos que sean los más representativos del resto. Estos reciben el nombre de “*exemplar*” (ejemplares), y cada uno de los demás elementos queda vinculado a uno de estos exemplars, el que mejor lo represente. Esta identificación que debe hacer no parte de una selección aleatoria que luego se refina, sino que considera simultáneamente todos los elementos del conjunto de datos como posibles exemplars, y a través de un proceso iterativo, va descartando gradualmente a los menos idóneos para ejercer el papel de exemplars del resto.

Para su funcionamiento el algoritmo necesita de dos elementos, el primero es una matriz de similitudes entre pares de elementos del conjunto de datos, en donde cada valor indica qué tan adecuado sería un elemento para actuar como exemplar de otro. El segundo es un valor llamado “*preference*” (preferencia), este es asignado a cada elemento del conjunto e indica qué tan deseable es que este sea seleccionado como un exemplar. Este último valor es de gran importancia debido a que influye en la cantidad de clústeres que se crearan, si se seleccionan valores muy altos de preferencia, entonces el algoritmo tendrá que identificar a una mayor cantidad de exemplars, por ende, más clústeres. Mientras que, si se seleccionan valores bajos de preferencia, habrá menos clústeres. Si no hay información previa que indique qué elementos son más idóneos para ser exemplars, entonces los autores recomiendan que se asigne un valor de preferencia común a todos los elementos, como por ejemplo la mediana de todas las similitudes de la matriz de entrada. Este algoritmo determina la cantidad de clústeres mediante su propio proceso de funcionamiento en base a los valores ya mencionados, no necesita que se le administre esta cantidad como parámetro previo a su ejecución (Frey y Dueck, 2007).

Una vez administrado con los valores necesarios, el funcionamiento del algoritmo para determinar exemplars se basa en un proceso iterativo de intercambio de mensajes entre pares de elementos. Existen dos tipos de mensajes, el primero se denomina “*responsibility*” (responsabilidad), este es un mensaje que un elemento le envía a otro indicándole qué tan adecuado sería ese candidato para representarlo, considerando la existencia de otros elementos que podrían cumplir ese rol. El segundo tipo se denomina “*availability*” (disponibilidad) este es un mensaje que uno de los elementos candidatos a exemplar, le envía a otro indicándole qué tan adecuado sería que ese elemento lo eligiera para cumplir tal rol, tomando en consideración

el apoyo que está recibiendo de otros elementos del conjunto. Estos mensajes se actualizan de forma iterativa, en cada iteración, los mensajes de responsabilidad se recalculan considerando las disponibilidades actuales y viceversa. Con cada iteración, la evidencia acumulada de los mensajes se va refinando hasta que surge un conjunto estable de exemplars. El algoritmo se detiene cuando los elementos seleccionados como exemplars dejan de cambiar durante un número determinado de iteraciones (Frey y Dueck, 2007).

Sin embargo, en base a la descripción del funcionamiento se reconocen dos limitaciones que posee este algoritmo. La primera es la necesidad de determinar el valor de preferencia, ya que este influye directamente en cuántos clústeres se generarán, una elección inadecuada puede resultar en la generación de un número de clústeres que no sea representativo de la estructura del conjunto de datos. La segunda limitación es la necesidad de proveerle al algoritmo la matriz completa de similitudes entre todos los pares de elementos, lo cual implica un costo de almacenamiento y procesamiento que aumenta considerablemente a medida que aumenta el tamaño del conjunto de datos. Esto podría resultar ser una gran limitación para nuestro caso en particular ya que se busca la clusterización de glosarios LEL voluminosos.

Clustering basado en distribución.

Uno de los enfoques más famosos para realizar un clustering basado en distribución es el conocido como *Gaussian Mixture Model* (GMM). Este enfoque es un modelo probabilístico que supone que los datos que tiene que procesar fueron generados por distintas fuentes con distribución gaussiana o normal, de ahí su nombre característico. Estas fuentes se combinan mediante coeficientes de mezcla, estos indican qué proporción de datos proviene de cada fuente. Este método permite construir modelos de densidad variable, ya que distintas regiones del espacio de datos pueden ser explicadas por distintos componentes de mezcla (Bishop, 2006).

Según Bishop (2006) para que este modelo pueda trabajar, hace uso de tres elementos: una media, una matriz de covarianza y un coeficiente de mezcla por cada componente. El primer elemento mencionado nos indica la zona central de un componente, en este caso un componente puede entenderse como un clúster probabilístico, la matriz de covarianza describe qué tan dispersos están los datos alrededor de este centro, junto con su forma y orientación, por último, el coeficiente de mezcla representa la proporción que ese componente representa en el modelo total, o dicho de otra forma que tan grande es. El autor también destaca que una de las características más importantes de este algoritmo son las variables latentes u ocultas, estas variables no se pueden observar, pero permiten estimar qué componente podría haber generado cada dato dentro del conjunto. A partir de esta idea surge el concepto de responsabilidad que

indica qué probabilidad hay de que un componente específico generara un elemento del conjunto de datos, es decir si el elemento está mejor explicado por un componente que los demás, este primer componente tendrá la probabilidad más alta de haberlo generado.

Estos parámetros descritos se estiman mediante un algoritmo llamado “Expectation-Maximization” (EM), este es un proceso iterativo que consta de dos pasos. En el paso E (Expectation) se calculan las responsabilidades de cada componente sobre cada elemento usando los parámetros actuales. En el paso M (Maximization) se actualizan las medias, covarianzas y coeficientes de mezcla a partir de esas responsabilidades. Este ciclo se repite hasta que se alcanza una estabilización en estos parámetros (Bishop, 2006).

Según el autor Bishop (2006) la diferencia con métodos basados en centroides como el K-Means es que en este cada elemento queda asignado a un grupo según qué centroide esté más próximo, provocando que cuando existe un elemento que se ubica aproximadamente entre dos centroides, la asignación de este a alguno de esos grupos genera incertidumbre sobre su pertenencia al mismo. Mientras que el GMM se centra en identificar distribuciones estimando qué distribución representa mejor a cada grupo y qué probabilidad tiene cada elemento de pertenecer a ese grupo, usando algo que se conoce como “soft assignment” (asignación blanda), en la cual cada elemento recibe un grado de pertenencia a cada grupo expresado en probabilidad en vez de asignación exclusiva.

Sin embargo, los GMM presentan algunas limitaciones que se deben tener en cuenta. En primer lugar, existe la variación de resultado según las variables iniciales elegidas, ya que dependiendo de estas el modelo puede variar bastante en sus resultados, por eso el autor Bishop (2006) menciona que lo mejor es utilizar primero un algoritmo como el del K-Means y luego en base a los resultados de este usarlos como punto de partida para GMM. La segunda es que necesita que se le indique de antemano el número de clústeres probabilísticos o componentes que debe buscar, lo cual no siempre es evidente cuando no se conoce la estructura de datos previamente. Por último, existe una probabilidad de que ocurra una singularidad, en la cual una de las componentes gaussianas colapsa sobre un único dato haciendo que reduzca su variabilidad a casi cero, lo que ocasiona que genere un resultado ficticio muy positivo. Ficticio porque esa componente gaussiana no está explicando un conjunto de datos sino uno en particular, el autor menciona que este es un fenómeno de sobreajuste severo.

4.1.3 Criterios para determinar número de clústeres

A lo largo de la sección previa, se mencionó que varios algoritmos que se utilizarán para este trabajo necesitan previo a su ejecución que se les indique la cantidad de clústeres que

se generarán. Y se debe entender que la elección de medidas de similitud y algoritmo de clusterización son cruciales, pero no suficientes para realizar las tareas de clusterización, también se debe tener en cuenta este elemento de cuántos grupos o clústeres se van a generar. Como también se ha mencionado esta es una decisión difícil puesto que no se puede saber cuántos clústeres son necesarios solo observando el conjunto de datos. Por este motivo se desarrollaron distintos métodos para poder estimar ese número de forma indirecta, evaluando las características de la agrupación generada.

El primero es el método del codo, este es uno de los métodos más famosos y se basa en que un buen agrupamiento debe minimizar la distancia entre los elementos de un mismo grupo. Para poder realizar esto hace uso de una medida llamada “Within-Cluster Sum of Squares” (WCSS), la cual representa la suma de distancias de cada elemento al centro del clúster al cual pertenecen. Esta técnica consiste en calcular la WCSS para distintos números de clústeres y realizar un gráfico con estos datos, el número óptimo de clústeres se corresponde con el punto en donde la gráfica cae de forma repentina y forma una curva con similitud a un codo (Alonso et al., 2025).

El segundo es el método de silueta promedio, este método combina dos aspectos para cada elemento del conjunto. El primero es la cohesión, esto es que tan similar es el elemento de los demás de su mismo grupo, el segundo es la separación la cual indica que tan distinto es el elemento de los del grupo más cercano. La silueta promedio varía su valor entre -1 y 1 esto es, los valores cercanos a 1 indican que el elemento analizado está bien ubicado, para los valores cercanos al 0 indican ambigüedad y para los cercanos al -1 indican que no está bien clasificado. Para poder seleccionar el número de clústeres óptimo, se realizan simulaciones con distintos valores de K (clústeres) y para cada uno de ellos se calcula la silueta promedio de conjunto de datos, seleccionándose aquel K que haga que la silueta promedio sea lo más cercana a 1 posible. Este método tiene la particularidad que a diferencia de los dos anteriores, con este se pueden comparar distintos algoritmos de clusterización (Alonso et al., 2025).

4.1.4 Métricas de evaluación internas de clustering

La evaluación del clustering generado, mediante métricas, es una tarea importante que debe llevarse a cabo por varios motivos planteados por Alonso et al. (2025). La primera de ellas es que en la clusterización no existe un valor real que indique la correcta pertenencia de los elementos a los grupos, contra el cual comparar lo obtenido por los métodos de clusterización, a diferencia de lo que ocurre con métodos supervisados. En otras palabras, no se puede saber si el resultado obtenido es adecuado o no. Además, construir un clúster implica la toma de

varias decisiones (medida de similitud, algoritmo, número de clústeres), las combinaciones entre estos pueden generar resultados muy distintos, por este motivo se debe contar con criterios objetivos que permitan comparar y poder así justificar la elección final. Como el objetivo de clusterizar se define en términos que pueden ser medibles como la maximización de distancia entre clústeres y la minimización de esta dentro de cada clúster, entonces la evaluación de calidad consiste justamente en cuantificar qué tan bien logra esos objetivos el resultado obtenido.

Coeficiente de silueta

Este método para la evaluación de los clústeres generados ya fue desarrollado en el apartado de criterios para determinar el número de clústeres, y se desarrolló como método de silueta promedio. El principio sobre el que se basa es exactamente el mismo, sin embargo, también puede servir para la evaluación de la calidad de los clústeres resultantes. Esto se debe a que, en su definición, combina los dos aspectos que caracterizan un buen clustering, el primero es la cohesión entre los elementos de cada grupo, y el segundo es la separación entre los grupos generados. Sumado a esto, al tener la particularidad de poder comparar algoritmos, como también se mencionó en el apartado de criterios para determinar el número de clústeres, se puede hacer uso de este método no solo para conocer la calidad del agrupamiento de cada técnica, sino también para determinar cuál obtuvo el mejor resultado: el algoritmo que produzca la silueta promedio más alta será el que generó el mejor agrupamiento.

Índice de Caliński-Harabasz

El índice de Caliński-Harabasz fue originalmente propuesto por Caliński y Harabasz (1974) con el nombre de “*Variance Ratio Criterion*” (VRC), esta es una medida que se basa en la división entre la dispersión entre los clústeres (que tan separados están) y la dispersión dentro de ellos (que tan compactados están los puntos dentro de ellos). Los autores habían propuesto este índice como un criterio informal para saber cuál es la cantidad óptima de clústeres, eligiendo aquella para la cual el índice alcanzara el máximo posible.

En la actualidad, este índice se usa también como métrica para evaluar la calidad de la clusterización generada, como lo presenta Scikit-learn (s. f.). En la documentación se menciona que un valor más alto de índice corresponde a un modelo con clústeres mejor definidos, siendo estos más densos internamente y están más separados unos de otros. Una de las ventajas que se resalta es la eficiencia computacional, lo cual lo hace adecuado para conjuntos de datos muy grandes. Y como una de sus limitaciones se menciona que el índice tiende a ser mayor en clústeres de forma convexa, por lo que se entiende que este penaliza aquellas formas que sean distintas, como los que se generan con algoritmos basados en densidad como DBSCAN.

Índice de Davies-Bouldin

El índice Davies-Bouldin propuesto por Davies y Bouldin (1979), es un tipo de índice que evalúa la calidad del agrupamiento a partir de la similitud entre pares de clústeres. Como en el anterior índice, el Calinski-Harabasz, este se basa en la combinación entre la dispersión interna del clúster y la externa entre clústeres. Y su funcionamiento particular es: para cada clúster se identifica cuál es más similar a este y luego se promedian esos valores para obtener una media global de la clusterización que fue realizada. Los autores mencionan que el agrupamiento que pueda minimizar esa medida será considerado como el mejor debido a que este indica que incluso con el par más problemático de clústeres está suficientemente diferenciado en relación con las dispersiones internas que poseen.

A nivel operativo Davies y Bouldin (1979) plantean que para identificar la similitud entre clústeres se utiliza un cálculo considerando la dispersión interna de ambos y la distancia entre los vectores que los representan, o en otras palabras sus centroides. En la documentación de Scikit-learn (s. f.) se menciona que el valor mínimo posible de este índice es 0 y que valores más cercanos a este mínimo representan un mejor resultado de la clusterización, puesto que reflejan clústeres más separados entre sí con mayor densidad de puntos dentro de ellos. Como ventajas se menciona que el cálculo es más sencillo que el utilizado por el índice silueta, y que se basa en características propias del conjunto de datos, ya que solo utiliza distancias entre puntos. Como limitaciones se destaca que penaliza a los algoritmos basados en densidad.

Métricas de evaluación externas de clustering

Las métricas de evaluación externas permiten analizar la calidad de la clusterización realizada. Para esto necesitan de una clasificación de referencia contra la cual compararla, con el objetivo de determinar cuál es el grado de similitud entre ambas, a esta referencia se la conoce comúnmente como *ground truth* (Vinh et al., 2010).

Índice Rand (RI) y Rand Ajustado (ARI)

El RI, propuesto por Rand (1971), es una de las medidas pertenecientes a la categoría de externas, este mide la similitud entre dos particiones de un mismo conjunto de objetos, es decir, toma el resultado de la clusterización como primera partición y el resultado provisto para su comparación como segunda partición. La idea central que maneja es que comparar elemento por elemento no es informativo, porque los nombres asignados a cada clúster son arbitrarios y dos resultados de agrupamientos que resulten equivalentes podrían parecer distintos solo por usar etiquetas diferentes. Entonces en su lugar RI analiza todos los pares posibles de elementos de una partición y corrobora que estos hayan sido tratados de la misma manera en la otra partición, que pertenezcan al mismo clúster en las dos particiones o que pertenezcan a clústeres

distintos en las dos. La magnitud de similitud entre ambas particiones es representada a través de un índice que puede tomar valores en el rango de $[0, 1]$ donde 1 representa el acuerdo perfecto entre ambas particiones, es decir, que todos los pares de valores posibles fueron agrupados de la misma manera en ambas particiones comparadas. Y el 0 representa que las dos particiones no tienen ninguna similitud.

Una de las limitaciones que tiene este índice es que su valor no es cero o cercano a esto cuando las dos particiones son generadas aleatoriamente, sino que por el contrario el índice podría tomar un valor relativamente alto solo por coincidencias fortuitas. Para solucionar esta limitación Hubert y Arabie (1985) propusieron el Adjusted Rand Index (ARI), esta es una revisión del índice anterior la cual descuenta del índice la porción de acuerdo que es atribuible al azar. La lógica detrás de este ajuste es que se calcula cual sería el acuerdo que se esperaría si las clusterizaciones fueran generadas de forma aleatoria (un valor de referencia) y luego se le resta ese valor al índice calculado, de esta forma se queda solo lo que excede a lo esperado por la aleatoriedad y se reescala el resultado de modo que el máximo (acuerdo perfecto) siga siendo 1. De esta forma, el ARI toma valores de 1 cuando las dos clusterizaciones coinciden a la perfección y 0 cuando la similitud entre ellas es equivalente a lo esperado por azar, y puede llegar a tomar valores negativos si el acuerdo resultante es incluso menor que el esperado por azar.

Según la documentación de Scikit-learn (s. f.) el rango de RI puede ir desde $[0, 1]$ mientras que el de ARI puede tomar valores entre $[-0,5; 1]$ siendo los valores más bajos representación de etiquetados independientes (muy distintos o coincidencias por azar) y los valores más altos representan el acuerdo perfecto. Entre las ventajas se destaca la interpretabilidad que posee, su rango acotado y el hecho de no asumir ninguna estructura particular en los clústeres, lo que permite utilizarlo para comparar resultados de algoritmos muy distintos entre sí. En cuanto a las limitaciones se menciona que estas métricas necesitan conocer la verdadera clasificación de los datos para poder compararla.

Normalized Mutual Information (NMI) y Adjusted Mutual Information (AMI)

La NMI es una métrica de evaluación que pertenece a la familia que está basada en la teoría de la información. El punto de partida de esta es la Mutual Information (MI), la cual es una medida que cuantifica que tanto se reduce la incertidumbre sobre una de las particiones cuando se conoce la otra o, dicho en otras palabras, que tan similares son. Si los dos coinciden totalmente, son iguales, entonces conocer uno permite reconstruir el otro y por ende la MI alcanza su máximo, si son totalmente distintos conocer uno no aporta ninguna información acerca del otro por ende la MI es muy cercana a 0. La NMI surge porque la MI no está acotada

en la parte superior del rango de valores que puede tomar, siendo este dependiente totalmente de las entropías presentes en las clusterizaciones comparadas. Para resolverlo, la NMI divide la MI por alguna combinación de las entropías de ambas particiones (típicamente su máximo, su mínimo, su promedio o la raíz de su producto), así de esta forma el valor que puede tomar queda acotado entre $[0, 1]$ donde 1 indica que las particiones son idénticas mientras que el 0 indica que son totalmente diferentes (Vinh et al., 2010).

Una importante limitación de la NMI es que incluso cuando se comparan dos agrupamientos generados al azar, estas no necesariamente dan como resultado un valor cercano a 0, como cabría esperar, y de hecho no solo eso, sino que el resultado tiende a incrementarse cuando el número de clústeres aumenta, lo que hace que la métrica esté sesgada particularmente con agrupaciones que presenten más clústeres. Para solucionar esto se desarrolló la Adjusted Mutual Information (AMI), una versión que garantiza que en caso de compararse dos agrupamientos generados al azar el valor producido será cercano a 0. La corrección sigue la misma lógica que la utilizada para el ARI: se calcula la diferencia entre el valor observado y el esperado por azar, y se la normaliza para que el máximo siga siendo 1 (Vinh et al., 2010).

La documentación de Scikit-learn (s. f.) menciona que una de las propiedades de AMI es que es simétrica, lo que significa que es útil incluso sin tener una clasificación verdadera y se busca contrastar los resultados de dos algoritmos distintos. También se menciona que sus ventajas son que un etiquetado perfecto recibe puntaje de 1 y que está preparada para tratar con asignaciones aleatorias, las cuales reciben un puntaje muy cercano a 0 sin importar el número de clústeres del conjunto de datos. En cuanto a su limitación se destaca que requiere conocer la clasificación verdadera de los datos, lo cual que en la práctica raramente está disponible.

4.2 Determinación de número de clústeres en experimento

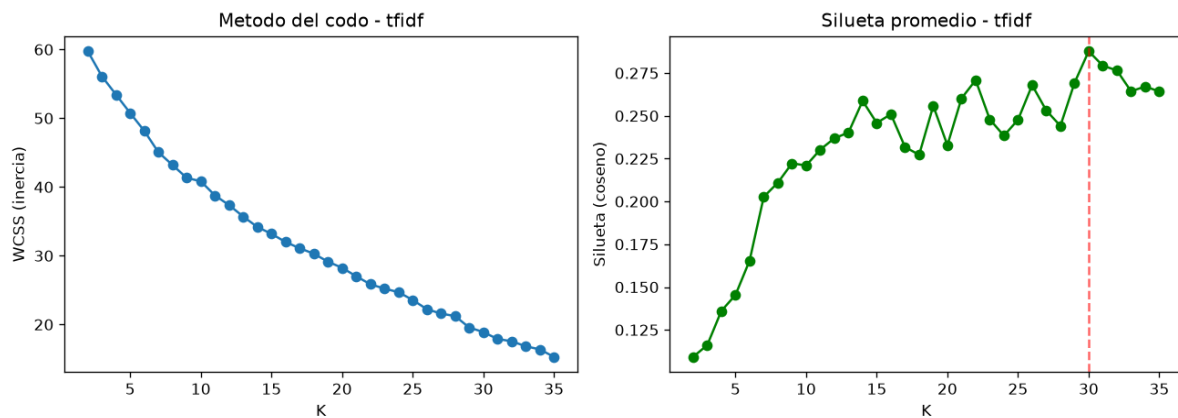
Como se menciona anteriormente, varios algoritmos de clustering necesitan que se les proporcione el número de clústeres para poder ejecutarse, en el caso de este trabajo son tres de los cinco algoritmos usados: K-Means, el agrupamiento jerárquico aglomerativo y el modelo de mezcla de gaussianas (GMM). Inicialmente, para calcular este valor se consideró la posibilidad de usar criterios internos, como el método del codo y el coeficiente silueta. Sin embargo, durante las pruebas experimentales se observó que estos métodos tendían a sobrefragmentar el conjunto de símbolos, seleccionando valores de K demasiado altos que no eran compatibles con la noción de subdominio funcional. Dicho de otra forma, si bien ciertos valores elevados tendían a mejorar los resultados matemáticos de la clusterización, se

generaban particiones demasiado pequeñas para ser interpretadas como áreas significativas del dominio.

En la Figura 1 se muestra cómo los métodos del codo y silueta, calculados para TF-IDF, escogen valores de K demasiado altos considerando la cantidad de símbolos con los que cuenta el glosario de pruebas. Esto en el método del codo llega a tal punto que ni siquiera adquiere su característica forma de codo, lo que refleja la ausencia de una separación natural entre grupos, mientras que el método de la silueta tiende a favorecer valores de K más altos. También un comportamiento similar fue visto en la selección de valores de K para SBERT, utilizando los mismos criterios.

Figura 1

Método del codo y silueta promedio calculada para TF-IDF



Nota. Elaboración propia. Los valores corresponden a la fase exploratoria de selección de K, calculados sobre la representación TF-IDF previa a la configuración final del experimento descrita en 4.2, por este motivo, no son directamente comparables con los de la Tabla 6. Su propósito es ilustrar la tendencia de ambos criterios a favorecer valores de K elevados, la cual también se observó con la configuración definitiva.

Es por el motivo antes expuesto que se tomó la decisión de fijar el valor de $K = 8$, en concordancia con la cantidad de subdominios definidos en el conjunto de referencia. Cabe la aclaración de que esta decisión solo afecta a los métodos que requieren que se les suministre K, mientras que para los demás que seleccionan este valor mediante sus propios procesos de ejecución, como DBSCAN y Affinity Propagation, mantienen esta característica intacta. Para el experimento de agrupamiento mediante LLM, se definió la cantidad de los grupos de modo

orientativo sugerido en sus instrucciones (entre 5 y 12). Esta decisión permite comparar de forma directa los grupos generados por los algoritmos con los identificados manualmente.

4.3 Configuración final del experimento

Una vez definido el número de clústeres para los algoritmos que lo requieren, se estableció la configuración final del experimento. A cada símbolo del glosario LEL se lo representó como una concatenación de su nombre, su noción y su impacto. Sobre estos textos se generaron las representaciones vectoriales de TF-IDF con preprocesamiento en español y se emplearon unigramas y bigramas, y para SBERT se generaron las representaciones mediante el modelo multilingüe paraphrase-multilingual-mpnet-base-v2. Ambas representaciones se normalizaron con norma L2 y se realizó una reducción de dimensionalidad mediante PCA conservando un 95% de la varianza. Este número se eligió como un compromiso entre conservar la mayor parte de la información presente en las representaciones vectoriales y reducir la dimensionalidad antes de aplicar los algoritmos de clustering. Valores menores podrían descartar información relevante, y los mayores reducirían menos la dimensionalidad y, por ende, disminuirían el efecto buscado por la reducción. En K-Means, el jerárquico aglomerativo y GMM se fijó $K = 8$, por concordancia con la cantidad de subdominios del conjunto de referencia, y GMM empleó covarianza diagonal por estabilidad numérica. Los parámetros de DBSCAN se determinaron por medio de un barrido sistemático (ϵ entre 0,10 y 0,90; MinPts entre 2 y 5), resultando $\epsilon = 0,55$ y MinPts = 2 para TF-IDF, y $\epsilon = 0,40$ y MinPts = 2 para SBERT. DBSCAN y Affinity Propagation determinaron por sí solos la cantidad de grupos (12 y 16 respectivamente con TF-IDF y para SBERT 15 y 14). El enfoque generativo, con LLM, empleó el modelo llama3.1 ejecutado localmente mediante Ollama, con las estrategias zero-shot y few-shot, y definió la cantidad de grupos dentro del rango orientativo indicado, siendo 6 en zero-shot y 5 en few-shot (los prompts completos se transcriben en el Anexo B).

4.4 Resultados obtenidos

En esta sección se presentarán los resultados al evaluar las diez combinaciones mediante técnicas internas y externas, siendo las internas las que miden cohesión y separación de clústeres y las externas las que comparan con el conjunto de referencia para saber que tal lo hizo el algoritmo en particular con respecto a mi criterio.

4.4.1 Métricas Internas

En la Tabla 6 podemos observar los valores obtenidos de cada una de las combinaciones de representaciones vectoriales y los algoritmos. Para la lectura se debe recordar que, en el coeficiente de silueta, cuyo rango va de -1 a 1 , un valor más alto indica un mejor agrupamiento, lo mismo para el índice de Calinski-Harabasz. Mientras que en el índice de Davies-Bouldin ocurre lo inverso, ya que su valor mínimo posible es 0 y los valores más cercanos a este reflejan un mejor resultado.

Tabla 6

Resultados de métricas internas de las distintas combinaciones

Representación	Algoritmo	Nº de Clústeres	Ruido	Silueta	Calinski_harabasz	Davies_bouldin
TF-IDF	K-Means	8	0	0,084	2,439	3,138
TF-IDF	Jerárquico	8	0	0,117	2,671	2,753
TF-IDF	GMM	8	0	0,051	2,116	3,162
TF-IDF	DBSCAN	12	51	0,499	3,930	0,913
TF-IDF	Affinity	16	0	0,138	2,279	2,110
SBERT	K-Means	8	0	0,226	6,246	2,060
SBERT	Jerárquico	8	0	0,210	5,812	1,972
SBERT	GMM	8	0	0,177	5,732	2,026
SBERT	DBSCAN	15	24	0,480	7,855	0,959
SBERT	Affinity	14	0	0,275	5,475	1,608

Nota: Elaboración propia.

A partir de estos resultados se puede observar que las combinaciones que se basan en SBERT presentan mejores valores internos que las que se basan en TF-IDF. Lo que muestra que la primera consigue una mayor separación de los grupos generados. Sin embargo, se debe destacar que los valores obtenidos no son lo suficientemente altos como para afirmar la existencia de clústeres separados claramente, sino que estos indican una estructura moderada.

Y para el caso de DBSCAN se deben tomar sus resultados con cautela porque este algoritmo puede, y de hecho lo hizo, dejar elementos sin asignar interpretándolos como ruido. Por este motivo, aunque algunos valores puedan parecer competitivos, estos son en base a lo

que se asignó, sin tener en cuenta lo que está clasificado como ruido. Por este motivo al interpretar estos valores se deben tener en cuenta junto al valor de ruido.

4.4.2 Métricas externas

En la Tabla 7 podemos observar los valores obtenidos de cada una de las combinaciones, estos valores indican qué tan similares fueron respecto al conjunto de referencia. Para su lectura se debe recordar que las dos están corregidas por azar: toman el valor 0 cuando la coincidencia entre las particiones es equivalente a la esperable por azar y 1 cuando la coincidencia es perfecta, también el ARI puede tomar, además, valores negativos cuando el acuerdo resulta incluso menor que el esperado por azar. Se optó por las versiones con esta corrección debido a que las versiones sin esta tienden a incrementarse a medida que aumentan los clústeres. Esta propiedad permite comparar de manera equitativa métodos que produjeron una cantidad distinta de grupos sin que esta métrica favorezca a los que generaron más.

Tabla 7

Resultados de métricas externas contra el conjunto de referencia

Representación	Algoritmo / Estrategia	ARI	AMI
TF-IDF	K-Means	0,194	0,358
TF-IDF	Jerárquico	0,207	0,354
TF-IDF	GMM	0,159	0,283
TF-IDF	DBSCAN	−0,032	0,029
TF-IDF	Affinity	0,152	0,309
SBERT	K-Means	0,282	0,414
SBERT	Jerárquico	0,263	0,395
SBERT	GMM	0,229	0,340
SBERT	DBSCAN	0,140	0,308
SBERT	Affinity	0,212	0,376
<i>Enfoque generativo (LLM)</i>			
—	Zero-shot	0,337	0,501
—	Few-shot	0,301	0,396

Nota. Elaboración propia. Los valores en negrita corresponden al mejor resultado de cada métrica.

El símbolo “—” en la columna representación, en el último segmento de la tabla, indica que el enfoque generativo no emplea una representación vectorial. Y sus métricas externas se calculan comparando directamente la partición generada contra el conjunto de referencia, por esto resultan independientes de la representación. Además, se presenta de forma diferenciada

puesto que constituye una familia distinta a la del clustering vectorial, ya que no genera un espacio vectorial sobre el cual calcular métricas internas, razón por la cual no figura en la Tabla 6 y sus resultados se reportan únicamente en las métricas externas. También cabe destacar que para este cálculo de AMI/ARI, los símbolos que fueron descartados como ruido por DBSCAN y los que no fueron asignados por el LLM se trataron como un grupo adicional.

4.5 Comparación entre representaciones

Al ver las tablas de la sección anterior y enfocarse en los resultados de las representaciones, se puede notar una sostenida superioridad del método SBERT sobre TF-IDF. Esta ventaja no se limita a un promedio ni es algo puntual de un solo algoritmo, sino que se repite en la totalidad de las combinaciones evaluadas, SBERT obtiene valores más altos que TF-IDF tanto en ARI como en AMI, en cada uno de los algoritmos. Esta consistencia resulta más significativa que el tamaño de las diferencias, debido a que descarta que la ventaja se deba al comportamiento de un algoritmo puntual, demostrando así que proviene de la representación misma.

En los algoritmos de K-Means, jerárquico aglomerativo, GMM y Affinity Propagation, la ventaja de SBERT es moderada, y esta se sitúa en torno a los 0,04 y 0,07 puntos de AMI respecto a su par TF-IDF. En el lado de TF-IDF se destaca la combinación con K-Means cuyo AMI alcanza los 0,358, sin embargo, queda por detrás de su equivalente con SBERT la cual alcanza un valor de 0,414. Esta misma tendencia se puede ver en ARI, donde la misma combinación (SBERT + K-Means) alcanza los 0,282 puntos frente a los 0,194 de su equivalente en TF-IDF.

En el caso particular de DBSCAN tenemos el mayor salto producido por el cambio de representación entre los algoritmos. Con TF-IDF, este algoritmo obtiene un AMI de apenas 0,029 y un ARI negativo ($-0,032$) lo que indica que la equivalencia con el grupo de referencia es incluso menor que si se hiciera aleatoriamente. Cuando reemplazamos la representación por SBERT, tenemos un salto significativo con un ARI de 0,140 y un AMI de 0,308. Sin embargo, aun con SBERT, DBSCAN sigue siendo el algoritmo de menor desempeño externo, por debajo de las demás configuraciones.

Finalmente, cabe destacar que esta ventaja no se manifiesta únicamente en las métricas externas. En los cuatro algoritmos de K-Means, jerárquico aglomerativo, GMM y Affinity Propagation las métricas internas muestran el mismo comportamiento, con SBERT alcanzan un coeficiente de silueta y un índice de Calinski-Harabasz más altos, y también un índice de Davies-Bouldin más bajo, que sus equivalentes con TF-IDF. La única excepción es DBSCAN,

en donde la versión con TF-IDF obtiene una silueta más alta y un Davies-Bouldin más bajo, este suceso encuentra su explicación en la cantidad de símbolos que el algoritmo descartó como ruido.

4.6 Comparación entre algoritmos

Habiendo quedado en evidencia la superioridad de SBERT se toma esta representación como base para esta comparativa. Lo que resulta más relevante en esta comparativa es que los algoritmos que asignan la totalidad de los símbolos (K-Means, jerárquico aglomerativo, GMM y Affinity Propagation) rinden de forma pareja y en un estrecho que se ubica, para AMI, entre 0,340 y 0,414 mientras que, para ARI, se ubica entre 0,212 y 0,282. Dentro del rango K-Means y el jerárquico se ubican levemente por encima mientras que GMM y Affinity Propagation ofrecen un rendimiento similar entre sí. Esta proximidad entre algoritmos de familias distintas demuestra que la elección del algoritmo incide en el resultado de forma mucho menos significativa que la elección de la representación vectorial.

Un comportamiento que merece un análisis particular es el de DBSCAN, debido a que dependiendo de qué métricas se mire las conclusiones que se obtienen son opuestas. Si miramos las internas este algoritmo sería el mejor de todos los analizados, obteniendo el coeficiente de silueta más alto con 0,499 para TF-IDF y 0,480 para SBERT, y a la vez el índice de Davies-Bouldin más bajo (0,913 y 0,959 respectivamente) considerablemente mejor que el resto de los algoritmos. Esto contrasta abruptamente con las métricas externas, las cuales lo califican como el peor de todos los algoritmos, dándole incluso un ARI negativo para TF-IDF.

La explicación para esta contradicción es la cantidad de elementos que el algoritmo descartó como ruido. Para TF-IDF, DBSCAN clasificó como ruido 51 de los 78 símbolos, y con SBERT, 24. Esto implica que el cálculo realizado por las métricas internas no se hizo sobre la totalidad de los símbolos sino sobre la minoría que sí logro agrupar el algoritmo. En otras palabras, el resultado de estas métricas no refleja una adecuada agrupación del glosario, sino que refleja la elevada cohesión de los pocos símbolos que, si fueron agrupados, luego de hacer a un lado los que quedaron etiquetados como ruido. Las métricas externas confirman esto, ya que, por dejar sin asignar a la mayoría de los símbolos, su coincidencia con el conjunto de referencia es la más baja. Este caso es evidencia suficiente para este trabajo de que analizar las métricas internas en solitario puede resultar engañoso.

También es relevante mencionar la cantidad de grupos que generaron los algoritmos que determinan ese número por sí solos. Tanto DBSCAN como Affinity Propagation generaron entre 12 y 16 grupos frente a los 8 del conjunto de referencia, lo que explica por qué los

algoritmos con un número de clústeres fijado en 8 lograron mayor coincidencia con el conjunto de referencia.

Por último, tenemos a la agrupación generada por el LLM, que obtuvo el mejor puntaje en las métricas externas. Esta pertenece a una familia distinta a la del clustering vectorial, ya que realiza el agrupamiento mediante razonamiento semántico y no a partir de distancias en un espacio vectorial, por este motivo carece de métricas internas.

4.7 Análisis de la matriz de contingencia

La matriz de contingencia permite analizar cómo se distribuyen los símbolos del glosario LEL procesado, mediante el cruce de cada subdominio real (filas) contra cada clúster generado (columnas). De este modo cada celda indica cuántos símbolos fueron asignados al clúster. Este apartado se centra en la mejor configuración de clustering, SBERT con K-Means, por ser la técnica que responde directamente a la primera pregunta de investigación. El enfoque con LLM, como pertenece a una familia distinta, se analiza en base a su propia matriz de contingencia en el siguiente capítulo, al evaluar la efectividad comparada de las técnicas.

Tabla 8

Matriz de contingencia: SBERT + K-Means vs. referencia

Subdominio (referencia)	C0	C1	C2	C3	C4	C5	C6	C7	Total
Gestión Extracciones	1	1			3			3	8
Gestión Pacientes			7		1				8
Gestión Sistema	1		4				5		10
Gestión Clínica		2			3				5
Gestión de Pagos	1	4		2	2				9
Gestión de Análisis Biotecnológico	3							6	9
Gestión de Consentimientos				7					7
Gestión Estudio	4	2	1	1	1	10		3	22
Total	10	9	12	10	10	10	5	12	78

Nota. Elaboración propia. Las columnas C0 a C7 corresponden a los ocho clústeres generados por el algoritmo. Las celdas vacías indican ausencia de símbolos.

La matriz muestra un paisaje heterogéneo, dado que algunos clústeres son muy cohesivos mientras otros están fragmentados. En el lado favorable tenemos dos clústeres

completamente puros: el C6 reúne cinco símbolos de Gestión Sistema correspondientes a operaciones genéricas (loguearse, cargar datos y archivos, generar PDF), y el C5 agrupa diez símbolos los cuales pertenecen a Gestión Estudio y conforman un eje temático bien definido, el ciclo de resultados del estudio (estudio esperando resultado, con resultado completo, entregado o retrasado, entre otros). El hecho de que el algoritmo haya sido capaz de aislar este conjunto sin mezclarlo demuestra que fue capaz de encontrar una subestructura dentro del subdominio más grande del glosario.

Por otro lado, también tenemos otros clústeres que, si bien no son puros, sí giran en torno a un eje semántico. Como ejemplo podemos mencionar al C3 que concentra los consentimientos, con los siete símbolos del subdominio, más un estado de Gestión Estudio que comparte la temática ("estudio esperando consentimiento firmado"), aunque incorpora también dos símbolos de pagos ajenos al eje, o el C7 que representa el eje biotecnológico (lotes, muestras y resultados asociados). Se debe notar que estos agrupamientos reúnen símbolos de distinta categoría según su significado, lo que indica que el algoritmo organizó los símbolos por su carga semántica y no por su rol sintáctico, el cual es un comportamiento esperado.

Sin embargo, se observa que la matriz da evidencias de ciertas limitaciones. Algunos subdominios se fragmentan sin encontrar un clúster propio, por ejemplo, Gestión Extracciones se dispersa en cuatro clústeres y Gestión Clínica se disuelve entre el C1 y el C4. Además, Gestión Pacientes y Gestión Sistema comparten el clúster C2, y el C4 agrupa mayoritariamente símbolos de tipo objeto (médicos, obra social, muestra) provenientes de subdominios distintos. Estas fragmentaciones son el motivo por el cual el valor de AMI resulta moderado, obteniendo (0,414) y confirman que la coincidencia no es total.

Para concluir, el análisis de la matriz muestra una recuperación del conjunto de referencia, parcial y dispareja. Dado que existen tanto clústeres que se dispersan sin un clúster propio, como también clústeres cohesivos organizados en torno a ejes temáticos reconocibles, llegando a aislar subestructura interna dentro del subdominio más extenso del glosario, el de estudios.

Capítulo 5 - Discusión

Este capítulo interpreta los resultados presentados en el Capítulo 4 para dar respuestas a las preguntas de investigación y la hipótesis formuladas. Se organiza en cuatro apartados: los dos primeros responden, respectivamente, a cada una de las preguntas de investigación, el tercero evalúa la hipótesis y el cuarto expone las limitaciones del estudio.

5.1 Respuesta a la pregunta de investigación 1

La primera pregunta de investigación plantea si es posible utilizar técnicas de clustering para identificar los distintos subdominios presentes en un glosario LEL. A partir de los resultados obtenidos, se puede afirmar que sí, pero con un matiz: el clustering demostró ser capaz de recuperar estructura temática coherente, pero su valor no radica en un procedimiento totalmente automático, sino que radica en su uso como herramienta de asistencia para el trabajo de análisis.

En el apartado cuantitativo, las métricas externas respaldan esta afirmación. La mejor configuración de clustering (SBERT con K-Means) alcanzó un AMI de 0,414 y un ARI de 0,282, valores que se sitúan claramente por encima del azar, esto debido a que ambas métricas son sus versiones corregidas, como se mencionó en apartados anteriores. Esto indica que las particiones generadas no son arbitrarias, estas tienen una correspondencia real, aunque parcial, con los subdominios del conjunto de referencia. Por supuesto que no se espera una coincidencia perfecta, debido a que los límites entre subdominios de un mismo dominio son de por sí graduales, sumado a eso la referencia utilizada representa una organización razonable del material, mas no la única posible.

El hecho de que el clustering recupere subdominios definidos manualmente, aunque sea de forma parcial es la clave para la respuesta a la pregunta. Un caso que se podría mencionar es el del eje biotecnológico, el cual el algoritmo reconstruyó como un agrupamiento propio reuniendo la mayoría de sus símbolos, además, dentro del subdominio más extenso, el de estudios, pudo aislar el ciclo de resultados en un grupo cohesivo. Esto confirma que las técnicas pueden capturar estructura semántica real y no solo regularidades superficiales. Dado que el interés de este trabajo radica en un escenario donde el glosario LEL sea de un tamaño considerable donde subdividir el dominio a mano resulta impracticable. Es ahí donde el comportamiento que se observó en este trabajo adquiere un valor. Si el clustering puede, sobre

un dominio conocido, reconstruir subdominios validos entonces sobre un glosario sin partición previa puede proponer una organización inicial para que sea revisada y ajustada.

5.2 Respuesta a la pregunta de investigación 2

Esta segunda pregunta busca esclarecer cuál es la combinación más efectiva para la identificación de subdominios. Como se mencionó anteriormente los resultados muestran que el factor más determinante no es el algoritmo sino la representación vectorial, y en ese apartado SBERT superó a TF-IDF, por lo que la representación vectorial pesa más que la elección del algoritmo de agrupamiento. En cuanto a los algoritmos, la combinación de SBERT con K-Means fue la más efectiva, con los mejores valores de coincidencia externa.

No obstante, el mejor puntaje externo de todo el experimento no lo obtuvo un algoritmo de clustering, sino el enfoque generativo. El LLM en su variante zero-shot obtuvo un AMI de 0,501 y un ARI de 0,337, siendo este el mejor resultado de la métrica externa, mientras que su variante few-shot obtuvo un AMI de 0,396 y un ARI de 0,301, esto sugiere que la incorporación de ejemplos en el prompt no favoreció la tarea. Sin embargo, al interpretar estos resultados se debe tener cautela, dado que el LLM pertenece a una familia distinta y su matriz de contingencia muestra un comportamiento peculiar, en vez de diferenciar, tiende a consolidar, fusionando varios subdominios en grupos de gran tamaño.

Tabla 9

Matriz de contingencia: LLM (zero-shot) vs Referencia

Subdominio (referencia)	Pac.	G.pac.	Muest.	Pagos	Turnos	Estud.	S/A	Total
Gestión Sistema	7					3		10
Gestión Pacientes	8							8
Gestión Extracciones		1	6	1				8
Gestión Clínica					2	3		5
Gestión de Pagos		2		5		2		9
Gestión de Consentimientos		6				1		7
Gestión de Análisis Biotecnológico						9		9
Gestión Estudio			2			18	2	22
Total	15	9	8	6	2	36	2	78

Nota. Elaboración propia. Los nombres de las columnas corresponden a las etiquetas asignadas por el modelo: Pac. (Pacientes), G. pac. (Gestión de pacientes), Muest. (Muestras), Estud. (Estudios) y S/A (Sin asignar). Las celdas vacías indican ausencia de símbolos.

En consecuencia, no hay una técnica que sea la más efectiva de forma absoluta, sino que esto depende del criterio que se tome. Si este criterio es la mejor coincidencia con el grupo de referencia entonces el LLM en su configuración zero-shot es el ganador. Pero si el criterio es asistir a la identificación de subdominios entonces la combinación SBERT con K-Means resulta preferible, porque diferencia la estructura que el LLM tiende a aplanar.

5.3 Evaluación de la hipótesis

La hipótesis planteada sostiene que la aplicación de técnicas de clusterización avanzadas a los términos de un glosario LEL permite una agrupación semántica para identificar

distintos subdominios dentro del dominio de dicho glosario. En base a los datos arrojados por las métricas y al análisis realizado, la hipótesis puede considerarse como respaldada.

El agrupamiento resultó efectivamente semántico, dicho de otra forma, los grupos formados fueron en función del significado semántico de los símbolos y no a su categoría gramatical, puesto que se reunieron objetos, verbos y estados de una misma temática, como se observa en la matriz de contingencia. Y esta agrupación semántica permitió identificar subdominios reales, de forma parcial pero definitivamente por encima del puro azar, es por esto por lo que se puede afirmar que el vínculo entre clusterización, agrupación semántica e identificación de subdominios que plantea la hipótesis, queda verificado

5.4 Limitaciones

Los resultados de este trabajo deben interpretarse considerando una serie de limitaciones que se detallan a continuación.

En primer lugar, el estudio se realizó sobre un único glosario LEL, y este pertenece a un solo dominio, que fue lo que el tutor Antonelli proveyó para la realización de este trabajo. Entonces por tratarse de un estudio de tipo exploratorio, las conclusiones, particularmente la referida a la técnica más efectiva, están vinculadas a las características de este glosario en particular. Y para la generalización de estas conclusiones se necesitaría replicar este experimento sobre varios glosarios pertenecientes a distintos dominios.

En segundo lugar, el conjunto de referencia usado para la evaluación es solo una posible organización del dominio, la cual fue definida manualmente. Y al tratarse de una construcción que está basada en un criterio propio, podrían existir particiones alternativas igualmente válidas, por lo cual se remarca que estas métricas externas miden la coincidencia con esa organización particular y no con una verdad absoluta.

En tercer lugar, el experimento con el LLM se realizó con un único modelo de lenguaje de código abierto (Llama 3.1), ejecutado localmente mediante Ollama. No se evaluaron modelos comerciales de mayor escala, por lo cual el desempeño que se muestra en las métricas corresponde a un modelo en particular y no representa el máximo que podría alcanzarse usando esta clase de técnica.

En cuarto lugar, en el experimento de LLM se deben tener ciertas consideraciones a la hora de interpretar sus resultados, puesto que las instrucciones del prompt sugerían un rango orientativo de subdominios (entre 5 y 12), por lo cual la granularidad del modelo fue orientada y no completamente libre. Además, el modelo recibió una versión reducida de cada símbolo, con nombre y noción truncada, sin los impactos, esto debido a que en modelos acotados los

prompts muy largos degradan el seguimiento de instrucciones y aumentan el riesgo de que la respuesta JSON llegue incompleta. Por esta razón se priorizó la parte de mayor densidad semántica de cada símbolo.

Por último, debemos mencionar que la comparación entre algoritmos no es completamente simétrica, dado que tres de ellos operaron con un número de clústeres fijado en 8, mientras que los restantes trabajaron con un valor determinado por ellos mismos. Esto si bien fue una decisión metodológica justificada, se debe tener en cuenta a la hora de interpretar los resultados obtenidos.

Capítulo 6 - Conclusiones Generales

Este trabajo se propuso evaluar, mediante un estudio de caso exploratorio, si las técnicas de clustering permiten identificar los subdominios de un glosario LEL y qué combinación de representación vectorial y algoritmo resulta más efectiva para ello.

La conclusión a la que se llegó es que efectivamente el clustering es capaz de recuperar estructura temática de un glosario LEL, sin embargo, su valor no reside en la automatización completa de la tarea, sino que se ubica en la asistencia al trabajo de análisis. Las particiones que se obtuvieron coincidieron parcialmente con las de la referencia, y las técnicas pudieron reproducir por sí solas agrupamientos significativos del dominio lo que demuestra que las técnicas capturan estructura semántica real. Esto obtiene mayor relevancia cuando se considera el escenario que motivó este trabajo, el de glosarios de gran tamaño, en los cuales la partición manual resulta muy poco conveniente y donde una propuesta automática de organización, aunque imperfecta, se presenta como un punto de partida muy útil.

En cuanto a la técnica más efectiva, se descubrió que el factor determinante resultó ser la representación vectorial en vez del algoritmo, como se suponía en un inicio. La representación SBERT superó de forma consistente a TF-IDF, mientras que las diferencias entre algoritmos fueron menores, y la combinación de SBERT con K-Means se destacó por sobre las demás. En cuanto al enfoque generativo, con LLM, obtuvo un puntaje mejor en las métricas externas, pero mostró una tendencia a consolidar la estructura en grupos grandes en lugar de diferenciarla, por este motivo la elección de la mejor técnica queda atada al objetivo perseguido: la mayor coincidencia numérica o la capacidad de distinguir subdominios finos.

Por último, el trabajo presentó una observación metodológica, la cual fue que, las métricas internas pueden ser engañosas cuando se las analiza en solitario, como se demuestra en el caso de DBSCAN, el cual según estas métricas era el mejor de todos, a pesar de tener el peor desempeño real. Esto demuestra que se debe tener una referencia externa a la hora de evaluar los resultados y no confiar ciegamente en indicadores de cohesión geométrica.

6.1 Trabajo Futuro

A partir de los resultados obtenidos y de las limitaciones señaladas, se identifican las siguientes líneas de trabajo futuro.

La primera es replicar el enfoque sobre glosarios de mayor tamaño y que pertenezcan a distintos dominios. Esto permitiría generalizar los resultados y sobre todo poner a prueba la

utilidad de las técnicas en el escenario que motiva este trabajo de investigación, aquel en el cual la partición manual resulta muy inconveniente.

La segunda es explorar criterios para la determinación automática del número de subdominios. Los métodos internos considerados, el del codo y la silueta, tendieron hacia una clara sobrefragmentación, lo que obligó a fijar un número de clústeres de forma manual. La investigación de criterios que estimen ese valor de forma automática que sea compatible con el subdominio funcional, permitiría ahorrar esa intervención manual.

En tercer lugar, evaluar modelos de lenguaje (LLMs) de mayor escala. El experimento se realizó con un modelo abierto de tamaño acotado, por lo cual sería interesante replicar este trabajo con modelos de mayor capacidad, incluidos los comerciales. De esta forma, se podrá determinar si los resultados mejoran, y si se mantiene la tendencia a consolidar la estructura en grupos grandes como se pudo observar en este trabajo, o si este comportamiento se debe al modelo utilizado.

Para finalizar, la cuarta sería validar este enfoque con analistas. Puesto que la conclusión de este trabajo es que la técnica tiene valor en la asistencia del análisis, entonces resultaría pertinente analizar en qué medida las particiones propuestas ayudan al analista en su trabajo real, con el fin de conocer su utilidad más allá de las métricas de coincidencia.

Referencias

- Alonso, J. C., Hoyos, C. C., y Largo, M. F. (2025). *Una introducción a los modelos de clústering empleando R*. Editorial Universidad Icesi. <https://doi.org/10.18046/EUI/bda.h.6>
- Amazon Web Services. (s. f.). *¿Qué es un modelo de lenguaje grande (LLM)?* <https://aws.amazon.com/es/what-is/large-language-model/>
- Antonelli, L., Lezoche, M., y Delle Ville, J. (2024). Knowledge extraction from the LEL glossary using natural language processing. *TecnoLógicas*, 27(59), e2917. <https://doi.org/10.22430/22565337.2917>
- Arora, C., Sabetzadeh, M., Briand, L. C., y Zimmer, F. (2017). Automated extraction and clustering of requirements glossary terms. *IEEE Transactions on Software Engineering*, 43(10), 918–945. <https://doi.org/10.1109/TSE.2016.2635134>
- Babaei Giglou, H., D'Souza, J., Mihindukulasooriya, N., y Auer, S. (2025). LLMs4OL 2025 Overview: The 2nd Large Language Models for Ontology Learning Challenge. *Open Conference Proceedings*, 6, 2913. <https://doi.org/10.52825/ocp.v6i.2913>
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Caliński, T., y Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1), 1–27. <https://doi.org/10.1080/03610927408827101>
- Chang, I.-C., Yu, T.-K., Chang, Y.-J., y Yu, T.-Y. (2021). Applying text mining, clustering analysis, and Latent Dirichlet Allocation techniques for topic classification of environmental education journals. *Sustainability*, 13(19), 10856. <https://doi.org/10.3390/su131910856>
- Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., y Washizaki, H. (2025). Generative AI for requirements engineering: A systematic literature review. *Software: Practice and Experience*, 56(2), 141–170. <https://doi.org/10.1002/spe.70029>
- Davies, D. L., y Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 224–227. <https://doi.org/10.1109/TPAMI.1979.4766909>
- Delle Ville, J., Torres, D., Fernández, A., y Antonelli, L. (2024). An approach to cluster scenarios according to their similarity using natural language processing. En *Human-Computer Interaction (HCI-COLLAB 2023)* (Communications in Computer and Information Science, Vol. 1877). Springer. https://doi.org/10.1007/978-3-031-57982-0_5

- Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- Ester, M., Kriegel, H.-P., Sander, J., y Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. En *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 226–231). AAAI Press. <https://cdn.aaai.org/KDD/1996/KDD96-037.pdf>
- Firmenich, D., Antonelli, L., Pazos, B., Lozada, F., y Morales, L. (2025). Exploring LLMs for user story extraction from mockups. *28th Workshop on Requirements Engineering (WER2025)*. <https://werpapers.dimap.ufrn.br/proceedings/WER2025/wer202509.html>
- Frey, B. J., y Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315(5814), 972–976. <https://doi.org/10.1126/science.1136800>
- Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., y Tehrani, S. Y. (2025). Research directions for using LLM in software requirement engineering: A systematic review. *Frontiers in Computer Science*, 7, 1519437. https://www.researchgate.net/publication/390064162_Research_directions_for_using_LLM_in_software_requirement_engineering_a_systematic_review
- Holdsworth, J., y Stryker, C. (s. f.). *¿Qué es el PLN (procesamiento del lenguaje natural)?* IBM. <https://www.ibm.com/es-es/think/topics/natural-language-processing>
- Hubert, L., y Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1), 193–218. <https://doi.org/10.1007/BF01908075>
- Institute of Electrical and Electronics Engineers. (1990). *IEEE standard glossary of software engineering terminology* (IEEE Std 610.12-1990). IEEE. https://www.informatik.htw-dresden.de/~hauptman/SEI/IEEE_Standard_Glossary_of_Software_Engineering_Terminology%20.pdf
- IBM. (s. f.). *Clustering*. <https://www.ibm.com/es-es/think/topics/clustering>
- Jain, A. K., Murty, M. N., y Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3), 264–323. <https://doi.org/10.1145/331499.331504>
- Jurafsky, D., y Martin, J. H. (2008). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (2.^a ed.). Pearson Prentice Hall.

- Khalid, S., Rasheed, U., Zaman, U. K. U., Alfayad, M., Assiri, M., Butt, W. H., Humayun, M., y Niazi, M. (2025). Ensuring quality in software requirement engineering process: A comparative study. *Egyptian Informatics Journal*, 29, 100147. <https://doi.org/10.1016/j.eij.2025.100754>
- Kumar, B., Tiwari, U. K., Dobhal, D. C., y Negi, H. S. (2022). User story clustering using K-means algorithm in agile requirement engineering. En *2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)* (pp. 1–5). IEEE. <https://doi.org/10.1109/CISES54857.2022.9844390>
- Leite, J. C. S. P., y Franco, A. P. M. (1993). A strategy for conceptual model acquisition. En *Proceedings of the IEEE International Symposium on Requirements Engineering* (pp. 243–246). IEEE. <https://doi.org/10.1109/ISRE.1993.324851>
- Li, J., Garijo, D., y Poveda-Villalón, M. (2025). Large language models for ontology engineering: A systematic literature review. *Semantic Web Journal*, XX(X), 1–53. <https://www.semantic-web-journal.net/system/files/swj4001.pdf>
- Manning, C. D., Raghavan, P., y Schütze, H. (2008). *An Introduction to information retrieval*. Cambridge University Press. <https://nlp.stanford.edu/IR-book/>
- Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv. <https://doi.org/10.48550/arXiv.1301.3781>
- Miller, J. K., y Alexander, T. J. (2025). Human-interpretable clustering of short text using large language models. *Royal Society Open Science*, 12(1), Artículo 241692. <https://doi.org/10.1098/rsos.241692>
- Necula, S.-C., Dumitriu, F., y Greavu-Şerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), Artículo 2055. <https://doi.org/10.3390/electronics13112055>
- Nguyen, M. A. (2026). *Tracing the evolution of word embedding techniques in natural language processing*. arXiv. <https://doi.org/10.48550/arXiv.2603.13271>
- Pennington, J., Socher, R., y Manning, C. D. (2014). GloVe: Global vectors for word representation. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). <https://doi.org/10.3115/v1/D14-1162>
- Pérez, G., Mostaccio, C., Antonelli, L., y Maltempo, G. (2025). Evaluation of natural language processing models to measure similarity between scenarios written in Spanish. *Cadernos do IME - Série Informática*. <https://doi.org/10.12957/cadinf.2024.87935>

- Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336), 846–850. <https://doi.org/10.1080/01621459.1971.10482356>
- Reimers, N., y Gurevych, I. (2019). *Sentence-BERT: Sentence embeddings using Siamese BERT-networks*. arXiv. <https://doi.org/10.48550/arXiv.1908.10084>
- Roldán Valdiviezo, P. M. (2025). *Definición de una gramática para el Léxico Extendido del Lenguaje* [Tesis de maestría, Universidad Nacional de La Plata]. SEDICI. <http://sedici.unlp.edu.ar/handle/10915/189172>
- Scikit-learn. (s. f.). 2.3. *Clustering*. <https://scikit-learn.org/stable/modules/clustering.html>
- Sommerville, I. (2005). *Ingeniería del software* (7.^a ed.). Pearson Educación. <https://apps.utel.edu.mx/recursos/files/r161r/w24792w/IRADS/Sommerville.pdf>
- Tao, C., Shen, T., Gao, S., Zhang, J., Li, Z., Hua, K., Tao, Z., y Ma, S. (2024). *LLMs are also effective embedding models: An in-depth overview*. arXiv. <https://arxiv.org/abs/2412.12591>
- Umar, M. A., y Lano, K. (2024). Advances in automated support for requirements engineering: A systematic literature review. *Requirements Engineering*, 29(2), 177–207. <https://doi.org/10.1007/s00766-023-00411-0>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). *Attention is all you need*. arXiv. <https://arxiv.org/abs/1706.03762>
- Vinh, N. X., Epps, J., y Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11, 2837–2854. <https://jmlr.org/papers/v11/vinh10a.html>
- Winland, V. (s. f.). *Zero-shot versus few-shot prompting*. IBM. <https://www.ibm.com/think/topics/zero-shot-vs-few-shot-prompting>
- Zadenoori, M. A., Dąbrowski, J., Alhoshan, W., Zhao, L., y Ferrari, A. (2025). *Large language models (LLMs) for requirements engineering (RE): A systematic literature review*. arXiv. <https://doi.org/10.48550/arXiv.2509.11446>
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M., Chioasca, E.-V., y Batista-Navarro, R. T. (2021). *Natural language processing for requirements engineering: A systematic mapping study*. https://pure.manchester.ac.uk/ws/portalfiles/portal/183513868/2020_12_16_NLP4RE_Accepted.pdf

Anexos

Anexo A. Glosario de términos técnicos

A continuación, se listan los términos técnicos y anglicismos usados a lo largo del trabajo que no cuentan con una definición en el cuerpo del documento.

Common Crawl: Repositorio público y gratuito de páginas web recopiladas de internet, utilizado como fuente masiva de texto para el entrenamiento de modelos de lenguaje.

Corpus: Conjunto de textos reunido para su estudio o procesamiento.

Count Vectorizer: Técnica de vectorización que representa cada documento contando cuántas veces aparece cada palabra del vocabulario, sin ponderar su relevancia como lo hace TF-IDF.

Dataset: Conjunto de datos estructurado que se utiliza para entrenar, probar o evaluar algoritmos y modelos.

Doc2vec: Extensión de Word2Vec que genera un vector por documento completo, en lugar de un vector por cada palabra.

FastText: Modelo de embeddings estáticos que construye la representación de cada palabra a partir de fragmentos de caracteres (subpalabras), lo que le permite representar palabras que no vio durante su entrenamiento.

Fine-tuning (ajuste fino): Proceso de reentrenar un modelo pre-entrenado sobre datos de una tarea o dominio específico, con el fin de especializar su comportamiento.

GenAI (inteligencia artificial generativa): Rama de la inteligencia artificial orientada a producir contenido nuevo (texto, imágenes, código) a partir de instrucciones, en lugar de limitarse a clasificar o analizar contenido existente.

Ground truth: Clasificación de referencia contra la cual se compara el resultado de un algoritmo.

Hardcodeado: Término usado para referirse a un valor fijado directamente en el código fuente o en la base de datos, que no puede modificarse mediante configuración.

Kernel sentences (oraciones núcleo): Oraciones simples con estructura básica sujeto-verbo-objeto sobre las cuales se aplican reglas de extracción de conocimiento en el enfoque de Antonelli et al. (2024).

LDA (Latent Dirichlet Allocation): Modelo probabilístico de tópicos que representa cada documento como una mezcla de temas, y cada tema como una distribución de palabras.

Machine learning (aprendizaje automático): Rama de la inteligencia artificial cuyos algoritmos aprenden patrones a partir de los datos, sin ser programados explícitamente para cada caso particular.

N-grama: Secuencia de n palabras consecutivas de un texto: un unigrama es una palabra aislada y un bigrama, un par de palabras contiguas. Los bigramas permiten capturar expresiones compuestas del dominio como “consentimiento informado” o “comprobante de pago”.

NumPy: Biblioteca de Python para el cálculo numérico eficiente sobre vectores y matrices.

Odds: Razón de probabilidades: cociente entre la probabilidad de que un evento ocurra y la probabilidad de que no ocurra.

Ollama: Herramienta de código abierto que permite descargar y ejecutar modelos de lenguaje de gran escala de forma local, sin depender de servicios en la nube.

Pandas: Biblioteca de Python para la manipulación y el análisis de datos tabulares.

PCA (Principal Component Analysis): técnica de reducción de dimensionalidad que proyecta los datos sobre un número menor de dimensiones (componentes), conservando la mayor proporción posible de la varianza original.

Prompt tuning: Optimización sistemática de las instrucciones (prompts) que se le dan a un modelo de lenguaje para mejorar su desempeño en una tarea, sin reentrenar el modelo.

Scikit-learn: Biblioteca de Python que implementa algoritmos de aprendizaje automático, de ella provienen los cinco algoritmos de clustering y las métricas utilizadas en este trabajo.

SpaCy: Biblioteca de Python para el procesamiento de lenguaje natural.

Stakeholders (partes interesadas): Personas u organizaciones que afectan o se ven afectadas por el sistema en desarrollo: clientes, usuarios finales, desarrolladores, entre otros.

SUS (System Usability Scale): Cuestionario estandarizado de diez preguntas empleado para medir la usabilidad percibida de un sistema.

Tester: Persona encargada de ejecutar pruebas sobre el software con el objetivo de detectar defectos antes de su entrega.

Anexo B. Prompts utilizados en el experimento con el LLM

Se transcriben los prompts empleados en el experimento por el LLM. En ambos casos el modelo recibió, además de las instrucciones, la lista completa de los 78 símbolos del glosario, que se abrevia como [LISTA DE LOS 78 IDENTIFICADORES] por razones de espacio y se le solicitó la respuesta en formato JSON. Los prompts carecen de tildes debido a que se transcriben tal cual como fueron enviados al modelo.

B.1 Reglas comunes a ambas estrategias

REGLAS IMPORTANTES:

1. Primero defini una lista de subdominios (campo "subdominios"). Tienen que ser POCOS (tipicamente entre 5 y 12) y representar areas funcionales del negocio (por ejemplo: Pagos, Muestras, Turnos, Gestion de pacientes, etc.).
2. Luego, en "asignacion", asigna CADA identificador a UNO de los subdominios que definiste. El valor DEBE ser exactamente uno de los nombres de la lista "subdominios". Esta PROHIBIDO inventar un nombre nuevo en la asignacion.
3. Varios simbolos relacionados con la misma actividad DEBEN compartir el mismo subdominio. No crees un subdominio por simbolo.
4. Tienes que asignar TODOS los identificadores, cada uno una sola vez.
5. Responde UNICAMENTE con el JSON, sin texto adicional.

B.2 Prompt de la estrategia zero-shot

Sos un analista experto en ingenieria de requerimientos. Abajo hay 78 simbolos de un glosario LEL de un sistema de laboratorio de analisis clinicos.

Cada simbolo tiene un identificador entre corchetes, un nombre y una descripcion.

Tu tarea es agruparlos en subdominios segun su area funcional o tematica.

[REGLAS COMUNES - ver B.1]

Formato exacto de la respuesta:

```
{"subdominios": ["Subdominio A", "Subdominio B"],
  "asignacion": {"SUJ-01": "Subdominio A", "OBJ-01": "Subdominio B"}}
```

Identificadores a asignar: [LISTA DE LOS 78 IDENTIFICADORES]

Lista de simbolos:

[LISTA DE LOS 78 SIMBOLOS: identificador, nombre y nocion]

B.3 Prompt de la estrategia few-shot

Sos un analista experto en ingeniería de requerimientos. Tu tarea es agrupar símbolos de un glosario LEL en subdominios según su área funcional.

EJEMPLO (de otro dominio, una veterinaria):

Simbolos:

[E1] Duenio: persona que trae a su mascota
 [E2] registrar duenio: dar de alta un duenio
 [E3] Vacuna: dosis que se aplica a un animal
 [E4] aplicar vacuna: administrar una dosis

Respuesta correcta:

```
{"subdominios": ["Gestion de duenios", "Vacunacion"],
  "asignacion": {"E1": "Gestion de duenios", "E2": "Gestion de duenios",
                 "E3": "Vacunacion", "E4": "Vacunacion"}}
```

Nota como [E1] (sujeto) y [E2] (verbo) comparten subdominio por pertenecer a la misma actividad, y como los nombres de la asignacion salen de la lista declarada.

Ahora hace lo mismo con esta lista real de un sistema de laboratorio de analisis clinicos.

[REGLAS COMUNES - ver B.1]

Identificadores a asignar: [LISTA DE LOS 78 IDENTIFICADORES]

Lista de simbolos:

[LISTA DE LOS 78 SIMBOLOS: identificador, nombre y nocion]

Anexo C. Conjunto de referencia

Se presenta el listado completo de los 78 símbolos del glosario LEL junto con su categoría y el subdominio al que fueron asignados en el conjunto de referencia.

Tabla C1

Símbolos del glosario LEL y subdominio asignado

Identificador	Nombre del símbolo	Categoría	Subdominio (referencia)
SUJ-01	Empleado	Sujeto	Gestión Sistema
SUJ-02	Admin / Administrador	Sujeto	Gestión Sistema
SUJ-03	Paciente	Sujeto	Gestión Pacientes
SUJ-04	Tutor	Sujeto	Gestión Pacientes
SUJ-05	Usuario	Sujeto	Gestión Sistema
OBJ-01	Médico derivante	Objeto	Gestión Clínica
OBJ-02	Médico informante	Objeto	Gestión Clínica
OBJ-03	Extraccionista	Objeto	Gestión Extracciones
OBJ-04	persona que retira la muestra / delivery	Objeto	Gestión Extracciones
OBJ-05	Obra social / OS	Objeto	Gestión de Pagos
OBJ-06	Diagnóstico presuntivo / Patología	Objeto	Gestión Clínica
OBJ-07	Presupuesto / ex Factura	Objeto	Gestión de Pagos
OBJ-08	comprobante de pago	Objeto	Gestión de Pagos
OBJ-09	Consentimiento informado	Objeto	Gestión de Consentimientos
OBJ-10	Consentimiento informado firmado	Objeto	Gestión de Consentimientos
OBJ-11	Turno	Objeto	Gestión Clínica
OBJ-12	Muestra	Objeto	Gestión Extracciones
OBJ-13	contador de estudios en estado “Esperando lote de muestra para iniciar procesamiento biotecnológico”	Objeto	Gestión de Análisis Biotecnológico
OBJ-14	Lote	Objeto	Gestión de Análisis Biotecnológico
OBJ-15	Estudio	Objeto	Gestión Estudio
OBJ-16	Tipo de Estudio	Objeto	Gestión Estudio
VER-01	editar entidad	Verbo	Gestión Sistema
VER-02	cargar datos	Verbo	Gestión Sistema
VER-03	crear nuevo Paciente	Verbo	Gestión Pacientes
VER-04	editar Paciente	Verbo	Gestión Pacientes

Identificador	Nombre del símbolo	Categoría	Subdominio (referencia)
VER-05	registrar Paciente	Verbo	Gestión Pacientes
VER-06	loguearse al sistema	Verbo	Gestión Sistema
VER-07	ver lista de pacientes	Verbo	Gestión Pacientes
VER-08	ver paciente	Verbo	Gestión Pacientes
VER-09	buscar paciente	Verbo	Gestión Pacientes
VER-10	crear PDF	Verbo	Gestión Sistema
VER-11	bajar PDF	Verbo	Gestión Sistema
VER-12	cargar archivo	Verbo	Gestión Sistema
VER-13	crear nuevo Estudio	Verbo	Gestión Estudio
VER-14	editar Estudio	Verbo	Gestión Estudio
VER-15	ver estudios	Verbo	Gestión Estudio
VER-16	ver lista de estudios	Verbo	Gestión Estudio
VER-17	ver estudio (empleado)	Verbo	Gestión Estudio
VER-18	ver estudio (paciente)	Verbo	Gestión Estudio
VER-19	buscar estudio	Verbo	Gestión Estudio
VER-20	bajar presupuesto	Verbo	Gestión de Pagos
VER-21	cargar el comprobante de pago	Verbo	Gestión de Pagos
VER-22	anular estudio por falta de pago	Verbo	Gestión de Pagos
VER-23	bajar consentimiento informado	Verbo	Gestión de Consentimientos
VER-24	cargar consentimiento informado firmado	Verbo	Gestión de Consentimientos
VER-25	anular comprobante de pago	Verbo	Gestión de Pagos
VER-26	elegir turno	Verbo	Gestión Clínica
VER-27	anular toma de muestra	Verbo	Gestión Extracciones
VER-28	cargar muestra	Verbo	Gestión Extracciones
VER-29	retirar muestra	Verbo	Gestión Extracciones
VER-30	crear lote	Verbo	Gestión de Análisis Biotecnológico
VER-31	ver lotes	Verbo	Gestión de Análisis Biotecnológico
VER-32	ver lote	Verbo	Gestión de Análisis Biotecnológico
VER-33	cargar resultado biotecnológico	Verbo	Gestión de Análisis Biotecnológico
VER-34	cargar interpretación de resultado	Verbo	Gestión de Análisis Biotecnológico
VER-35	ver liquidación de extracciones	Verbo	Gestión Extracciones
VER-36	liquidar extracciones	Verbo	Gestión Extracciones
VER-37	ver reportes	Verbo	Gestión Sistema

Identificador	Nombre del símbolo	Categoría	Subdominio (referencia)
EST-01	Estudio Esperando comprobante de pago	Estado	Gestión de Pagos
EST-02	Estudio Anulado por falta de pago	Estado	Gestión de Pagos
EST-03	Enviar consentimiento informado	Estado	Gestión de Consentimientos
EST-04	Estudio Esperando consentimiento informado firmado	Estado	Gestión Estudio
EST-05	Estudio Esperando la selección de turno para la extracción de sangre	Estado	Gestión Estudio
EST-06	Estudio Esperando toma de muestra	Estado	Gestión Estudio
EST-07	Estudio Esperando retiro de muestra desde el sector de extracciones	Estado	Gestión Estudio
EST-08	Estudio Esperando lote de muestra para iniciar procesamiento biotecnológico	Estado	Gestión Estudio
EST-09	Lote En procesamiento	Estado	Gestión de Análisis Biotecnológico
EST-10	Lote Procesado	Estado	Gestión de Análisis Biotecnológico
EST-11	Estudio Esperando resultado biotecnológico	Estado	Gestión Estudio
EST-12	Estudio Esperando interpretación de resultados e informes	Estado	Gestión Estudio
EST-13	Estudio Esperando ser entregado al médico derivante	Estado	Gestión Estudio
EST-14	Estudio con Resultado entregado	Estado	Gestión Estudio
EST-15	Estudio Esperando resultado	Estado	Gestión Estudio
EST-16	Estudio con Resultado completo	Estado	Gestión Estudio
EST-17	Estudio No retrasado	Estado	Gestión Estudio
EST-18	Estudio Retrasado	Estado	Gestión Estudio
EST-19	Consentimiento informado No firmado	Estado	Gestión de Consentimientos
EST-20	Consentimiento informado Firmado	Estado	Gestión de Consentimientos

Nota. Elaboración propia. Los símbolos se presentan agrupados por categoría (sujetos, objetos, verbos y estados).

Anexo D. Barrido de parámetros de DBSCAN

Se documenta el procedimiento de selección de parámetros de DBSCAN mencionado en el apartado 4.2. Para cada representación vectorial se exploraron 68 combinaciones (ϵ entre 0,10 y 0,90 en pasos de 0,05, y MinPts entre 2 y 5), y se seleccionó la configuración que maximizó el producto entre la silueta promedio y la cobertura. Este criterio premia la separación de los grupos formados, pero penaliza las configuraciones que descartan muchos símbolos como ruido, y no utiliza el conjunto de referencia en ningún paso de la selección. La Tabla D1 presenta las tres mejores configuraciones obtenidas para cada representación.

Tabla D1

Mejores configuraciones de DBSCAN por representación

Representación	ϵ	MinPts	Grupos	Ruido	Cobertura	Silueta	Score
TF-IDF	0,55	2	12	51	34,6 %	0,499	0,173
	0,60	2	12	39	50,0 %	0,325	0,162
	0,65	2	12	32	59,0 %	0,269	0,159
SBERT	0,40	2	15	24	69,2 %	0,480	0,332
	0,35	2	14	32	59,0 %	0,541	0,319
	0,30	2	10	42	46,2 %	0,595	0,275

Nota. Elaboración propia. En negrita, la configuración seleccionada para cada representación. *Cobertura:* proporción de símbolos asignados a algún grupo (no descartados como ruido). *Silueta:* silueta promedio de los símbolos asignados, calculada con distancia coseno. *Score:* producto entre la silueta y la cobertura, criterio utilizado para la selección.