Revista esprint Investigación

https://rei.esprint.tech
Vol. 5 N° 2, julio-diciembre 2026 (67-93)

ISSN: 2960-8317

Artículo de revisión

Marco metodológico computacional para el análisis estructural de problemas matemáticos verbales

A computational methodological framework for the structural analysis of mathematical word problems

Lizbeth Carolina Sanunga Guananga*
Universidad Nacional de Chimborazo
Riobamba - Ecuador
lizbeth.sanunga@unach.edu.ec
https://orcid.org/0009-0004-6929-0285

Carlos Luis Gusqui Guananga
Universidad Técnica de Ambato
Ambato - Ecuador
gusquicarlos@gmail.com
https://orcid.org/0000-0001-5449-5967

María Cristina Robayo Villarroel
Universidad Nacional de Chimborazo
Riobamba - Ecuador
maria.robayo@unach.edu.ec
https://orcid.org/0009-0007-4746-2930

Bryan Fernando Pérez-Pilco
Universidad Nacional de Chimborazo
Riobamba - Ecuador
bryanf.perez@unach.edu.ec
https://orcid.org/0000-0003-0288-5829

Dayana Cristina Villarreal Meza
Universidad Nacional de Chimborazo
Riobamba - Ecuador
dayanac.villarreal@unach.edu.ec
https://orcid.org/0000-0002-6971-6950

***Correspondencia:**
lizbeth.sanunga@unach.edu.ec

Cómo citar este artículo:
Sanunga, L., Gusqui, C., Robayo, M., Pérez-Pilco, B., & Villarreal, D. (2026). Marco metodológico computacional para el análisis estructural de problemas matemáticos verbales. *Esprint Investigación, 5*(2), 67-93. https://doi.org/10.61347/ei.v5i2.350

Recibido: *3 de junio de 2026*
Aceptado: *10 de julio de 2026*
Publicado: *16 de julio de 2026*

Resumen: La traducción de problemas verbales a expresiones algebraicas constituye uno de los obstáculos cognitivos más persistentes en la educación matemática, evidenciado en evaluaciones de desempeño estudiantil en la región. Para abordar esta brecha, el presente estudio diseñó y validó un marco computacional teórico-metodológico para analizar la estructura semántica de problemas matemáticos verbales, integrando la teoría de registros de representación semiótica de Duval (2006, 2017) con técnicas de procesamiento de lenguaje natural asistidas por modelos de lenguaje de gran escala. La validación se realizó mediante implementación en Python aplicada a un corpus de 48 problemas extraídos de libros de texto oficiales de educación secundaria ecuatoriana (EGB superior y BGU). Los resultados mostraron que el pipeline de cuatro módulos clasifica las estructuras semánticas con un desempeño F1 = 0,88 y κ = 0,83, evidenciando alta concordancia con anotadores humanos. Como contribuciones principales, el estudio derivó una taxonomía de seis estructuras semánticas que permite caracterizar la complejidad de los problemas y un flujo metodológico replicable documentado en anexos de código y cálculos estadísticos. Esta aproximación proporciona herramientas diagnósticas útiles para docentes e investigadores, permite diseñar secuencias didácticas fundamentadas, identificar errores específicos en la conversión semiótica y equilibrar la representación de estructuras semánticas en los textos curriculares. Además, establece líneas futuras de investigación que incluyen la ampliación del corpus a otros contextos latinoamericanos, el desarrollo de un módulo de estimación de carga cognitiva y la implementación de intervenciones didácticas basadas en la taxonomía, evaluando su impacto en el rendimiento estudiantil.

Palabras clave: Análisis semántico, didáctica de la matemática, NLP, problemas verbales, registros de representación semiótica, taxonomía computacional.

Abstract: *The translation of verbal problems into algebraic expressions is one of the most persistent cognitive obstacles in mathematics education, as evidenced by student performance assessments in the region. To address this gap, the present study designed and validated a computational theoretical-methodological framework to analyze the semantic structure of mathematical word problems, integrating Duval's (2006, 2017) theory of semiotic representation registers with natural language processing techniques assisted by large language models. Validation was performed via Python implementation applied to a corpus of 48 problems extracted from official Ecuadorian secondary education textbooks (EGB Superior and BGU). Results showed that the four-module pipeline classified semantic structures with an F1 = 0.88 and κ = 0.83, demonstrating high agreement with human annotators. Key contributions include a taxonomy of six semantic structures enabling problem complexity characterization and a replicable methodological workflow documented in code and statistical annexes. This approach provides diagnostic tools for teachers and researchers, supports the design of informed instructional sequences, identifies specific errors in semiotic conversion, and balances the representation of semantic structures in curricular texts, while establishing future research directions to evaluate its impact on student performance.*

Keywords*: Computational taxonomy, mathematics didactics, NLP, semantic analysis, semiotic representation registers, word problems.*

1. Introducción

Los problemas matemáticos de enunciado, denominados en la literatura internacional *mathematical word problems* (MWPs), constituyen un recurso pedagógico fundamental para vincular el razonamiento algebraico con situaciones contextualizadas del mundo real. No obstante, la dificultad de estos problemas trasciende la aplicación de algoritmos matemáticos, debido a que exige interpretar información lingüística, establecer relaciones semánticas y transformar representaciones verbales en estructuras matemáticas formales (Verschaffel et al., 2000; Polya, 1957).

Diversas investigaciones evidencian que los estudiantes frecuentemente responden los problemas verbales mediante procedimientos rutinarios sin considerar el significado del contexto planteado. Este fenómeno, denominado "suspensión del sentido" (*suspension of sense-making*), refleja una desconexión entre la comprensión del lenguaje natural y la construcción del modelo matemático requerido para resolver el problema (Verschaffel et al., 2000).

Esta dificultad representa una brecha de procesamiento semántico que constituye un desafío tanto para la educación matemática como para el desarrollo de sistemas inteligentes orientados a la comprensión automática del lenguaje. Zhang et al. (2020) señalan que la interpretación semántica de los problemas matemáticos verbales continúa siendo uno de los principales obstáculos para los solucionadores automáticos basados en procesamiento de lenguaje natural (PLN).

Desde la perspectiva cognitiva, la comprensión matemática implica la coordinación entre diferentes registros de representación, donde el estudiante debe realizar conversiones entre el lenguaje natural, expresiones algebraicas, tablas y representaciones gráficas (Duval, 2006, 2018). En este sentido, Duval (2014) sostiene que el aprendizaje matemático depende de la capacidad de movilizar y transformar representaciones semióticas, debido a que la construcción del significado matemático no ocurre únicamente mediante operaciones formales, sino mediante procesos de interpretación y conversión entre sistemas representacionales.

En este contexto, los errores en la resolución de problemas no necesariamente reflejan ausencia de conocimiento matemático, sino dificultades para establecer correspondencias entre las estructuras lingüísticas del enunciado y los objetos matemáticos involucrados. Cascella et al. (2021) evidenciaron que modificaciones en la formulación de los problemas pueden influir en las respuestas de los estudiantes, demostrando que las características lingüísticas y estructurales del enunciado condicionan los procesos cognitivos empleados durante la resolución.

La evidencia empírica también muestra que la dimensión lingüística constituye un componente determinante del pensamiento algebraico. Sibgatullin et al. (2022), mediante una revisión sistemática sobre pensamiento algebraico en educación, identificaron que la comprensión de relaciones expresadas verbalmente representa uno de los factores que condicionan el desarrollo del razonamiento matemático. Por tanto, el análisis de la estructura lingüística de los problemas constituye una vía relevante para comprender las dificultades que enfrentan los estudiantes durante los procesos de modelización matemática.

En el contexto educativo ecuatoriano, los resultados obtenidos en evaluaciones nacionales e internacionales han evidenciado dificultades persistentes en la resolución de problemas matemáticos contextualizados. Estas limitaciones sugieren que parte de los errores pueden estar relacionados con procesos de interpretación del enunciado y no exclusivamente con el dominio de procedimientos matemáticos (Verschaffel et al., 2020). Además, Vicente et al. (2022) demostraron que determinadas características presentes en los libros de texto pueden favorecer estrategias mecánicas de resolución, limitando la construcción de modelos matemáticos basados en la comprensión del contexto.

Frente a esta problemática, el procesamiento de lenguaje natural surge como una alternativa metodológica para analizar automáticamente las características lingüísticas y semánticas de los problemas matemáticos verbales. Investigaciones recientes han empleado técnicas de aprendizaje automático y modelos lingüísticos para clasificar la complejidad textual, identificar estructuras matemáticas y generar retroalimentación automatizada (Bednorz & Kleine, 2023; Botelho et al., 2023).

A diferencia de los enfoques orientados exclusivamente a obtener la respuesta correcta de un problema matemático, este estudio propone utilizar técnicas de PLN para analizar y clasificar la estructura semántica de los enunciados matemáticos, con el propósito de generar información diagnóstica sobre la demanda cognitiva asociada a cada problema. De esta manera, se busca aportar herramientas que permitan comprender la relación entre complejidad lingüística, representación matemática y procesos de resolución.

El objetivo del presente estudio es analizar la estructura semántica de problemas matemáticos de enunciado mediante técnicas de procesamiento de lenguaje natural, con la finalidad de identificar patrones lingüísticos asociados a su complejidad cognitiva y contribuir al desarrollo de estrategias de apoyo para la enseñanza de las matemáticas.

2. Desarrollo

La brecha semántica en la resolución de problemas verbales

La investigación sobre la resolución de problemas matemáticos tiene sus fundamentos modernos en el trabajo de Polya (1957), quien estableció cuatro fases heurísticas: comprensión del problema, elaboración de un plan, ejecución y revisión. Desde esta perspectiva, la comprensión constituye el punto inicial del proceso resolutivo, debido a que permite interpretar la información disponible y establecer relaciones entre los datos y la incógnita planteada.

Décadas después, Verschaffel et al. (2000) ampliaron esta perspectiva al demostrar que los estudiantes pueden desarrollar una "suspensión del sentido" al resolver problemas verbales. Este fenómeno ocurre cuando los estudiantes responden mediante procedimientos rutinarios basados en señales superficiales del enunciado, como palabras clave asociadas a operaciones, sin construir una representación adecuada de la situación matemática planteada.

La revisión sistemática de Sibgatullin et al. (2022), basada en 36 estudios empíricos sobre pensamiento algebraico, evidencia que las principales dificultades se concentran en la fase de representación. Los estudiantes suelen ejecutar procedimientos algebraicos correctamente cuando reciben una ecuación previamente formulada; sin embargo, presentan limitaciones cuando deben construir dicha representación a partir de un texto verbal.

Vicente et al. (2022) demostraron que la estructura y diversidad de los problemas matemáticos verbales (*mathematical word problems*, MWPs) presentes en los materiales curriculares influyen en las estrategias de resolución desarrolladas por los estudiantes. Estos hallazgos justifican la necesidad de herramientas capaces de analizar la complejidad semántica de los enunciados antes de abordar únicamente su solución matemática.

Comprender un problema verbal implica más que interpretar un conjunto de palabras; requiere construir una representación mental de la situación que articule el conocimiento matemático con la información contextual descrita. Este proceso depende de factores lingüísticos y estructurales como el número de entidades involucradas, las relaciones matemáticas implícitas (aditivas, multiplicativas o

proporcionales), la presencia de información irrelevante y la densidad conceptual del vocabulario empleado (Verschaffel et al., 2000).

La teoría de registros de representación semiótica

Duval (2006, 2017) desarrolló la Teoría de los Registros de Representación Semiótica (TRRS) como un marco para explicar los procesos de conceptualización matemática. Desde esta teoría, el aprendizaje matemático requiere la coordinación entre diferentes sistemas de representación, debido a que los objetos matemáticos no son accesibles directamente, sino mediante representaciones semióticas.

Un registro semiótico constituye un sistema de representación que permite realizar tres tipos de operaciones: representación, tratamiento y conversión. El tratamiento corresponde a las transformaciones realizadas dentro del mismo registro, mientras que la conversión implica cambiar de un registro a otro manteniendo el mismo objeto matemático de referencia.

En los problemas matemáticos verbales, el enunciado activa principalmente el registro de lengua natural, mientras que la resolución requiere una conversión hacia un registro algebraico-simbólico. Duval (2014) señala que esta transición implica una actividad semio-cognitiva compleja, debido a que la comprensión matemática depende de la capacidad de reconocer un mismo objeto bajo diferentes formas de representación y establecer relaciones entre ellas.

Duval (2006) sostiene que las conversiones entre registros no son procesos isomorfos, debido a que las propiedades visibles en una representación pueden no mantenerse explícitamente en otra. Por esta razón, la transformación del lenguaje natural hacia expresiones matemáticas constituye uno de los principales obstáculos cognitivos en la resolución de problemas verbales.

Desde una perspectiva computacional, esta teoría permite fundamentar el análisis semántico de los MWPs, ya que la identificación automática de variables, constantes y relaciones matemáticas presentes en el lenguaje natural puede aproximarse al proceso de conversión semiótica requerido para resolver cada problema. En consecuencia, el análisis mediante PLN puede contribuir a caracterizar la demanda cognitiva asociada a diferentes estructuras lingüísticas.

Somasundram (2021), mediante un modelo de ecuaciones estructurales aplicado a 720 estudiantes, identificó que el sentido simbólico constituye un predictor relevante del pensamiento algebraico. De forma complementaria, Ferretti et al. (2022) evidenciaron que las dificultades en las transformaciones semióticas explican problemas persistentes en la comprensión de expresiones algebraicas dentro de evaluaciones de gran escala.

Procesamiento de Lenguaje Natural aplicado a problemas matemáticos

Zhang et al. (2020) realizaron una revisión del desarrollo de los sistemas de procesamiento de lenguaje natural aplicados a problemas matemáticos verbales, desde aproximaciones basadas en reglas hasta modelos actuales de aprendizaje profundo. Los autores identifican la brecha de interpretación semántica como uno de los principales desafíos para lograr sistemas capaces de comprender la estructura matemática implícita en los textos.

Acharya et al. (2022) propusieron un sistema basado en clasificación y reglas lingüísticas para resolver problemas aritméticos verbales, alcanzando resultados relevantes en operaciones básicas. Estos avances evidencian que la combinación entre información lingüística y estructuras matemáticas explícitas puede mejorar la interpretación automática de los enunciados.

Un avance significativo corresponde al trabajo de Opedal et al. (2023), quienes desarrollaron MathWorld, un formalismo semántico basado en grafos que representa entidades, acciones y relaciones matemáticas presentes en problemas narrativos. Este enfoque resulta relevante porque aproxima la representación computacional del problema a una estructura similar al modelo mental que construye el estudiante durante la resolución.

Bednorz y Kleine (2023) demostraron que el análisis de características lingüísticas permite identificar dimensiones latentes de complejidad en problemas matemáticos verbales mediante aprendizaje automático no supervisado. Estos resultados respaldan el uso de técnicas de clasificación semántica orientadas a comprender la dificultad lingüística de los problemas.

Reimers y Gurevych (2020) desarrollaron modelos de representación semántica multilingüe mediante destilación de conocimiento, permitiendo generar embeddings con capacidad de comparación entre diferentes idiomas. Estas representaciones constituyen una herramienta relevante para el análisis automático de similitud y estructura semántica en corpus educativos multilingües.

Ughade y Kumbhar (2020) demostraron la aplicabilidad de técnicas de reconocimiento de entidades con nombre (*Named Entity Recognition*, NER) para identificar componentes relevantes dentro de problemas matemáticos escritos. De esta manera, el PLN proporciona mecanismos para extraer información lingüística necesaria para la construcción de modelos semánticos.

El desarrollo de los modelos de lenguaje de gran escala (*Large Language Models*, LLMs) ha ampliado las posibilidades del análisis automático del lenguaje matemático. Sin embargo, en este estudio estos modelos no se emplean como solucionadores automáticos, sino como herramientas de extracción semántica estructurada complementadas con reglas lingüísticas que favorecen la reproducibilidad y trazabilidad del análisis (Sundaram et al., 2024; Hwang & Utami, 2024).

3. Metodología

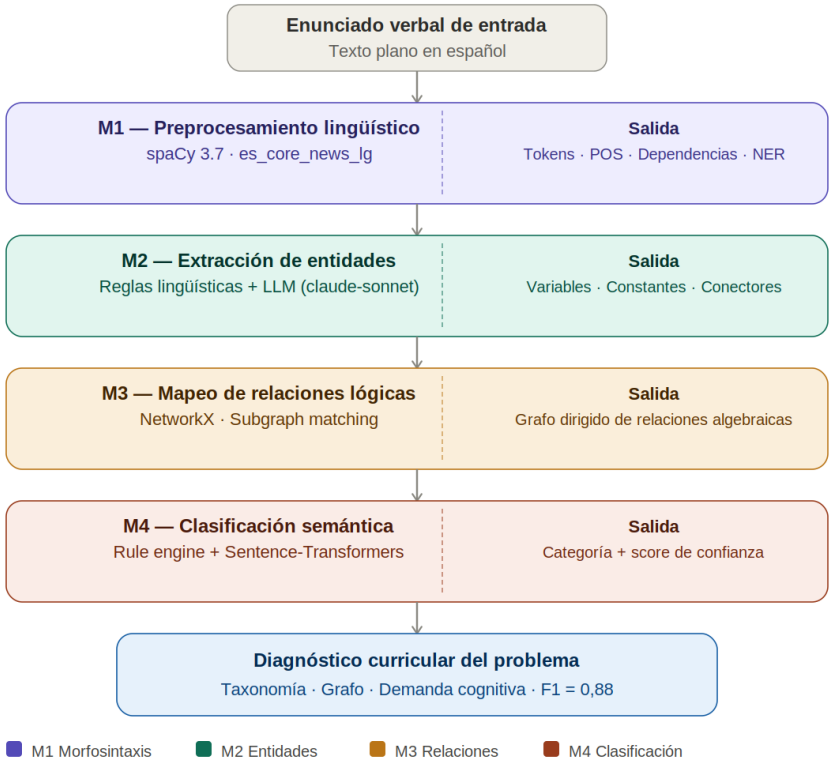
Diseño del estudio

El presente trabajo se desarrolló bajo un diseño teórico-metodológico con validación algorítmica orientado al análisis estructural de problemas matemáticos verbales mediante técnicas de procesamiento de lenguaje natural. La validación del marco propuesto se efectuó en dos dimensiones principales: (a) validez de contenido, mediante la verificación de que las categorías taxonómicas definidas cubrieran las estructuras semánticas presentes en los textos analizados; y (b) fiabilidad interanotador, mediante la comparación entre las clasificaciones generadas por el pipeline computacional y las realizadas manualmente por especialistas en didáctica de la matemática.

El proceso metodológico siguió el flujo de trabajo representado en la figura 1, compuesto por las fases de construcción del corpus, procesamiento lingüístico, extracción de entidades, modelización relacional y clasificación semántica.

Figura 1

Pipeline computacional de cuatro módulos para el análisis estructural de problemas verbales matemáticos



Constitución del corpus

El corpus se construyó a partir de los libros de texto oficiales del Ministerio de Educación del Ecuador correspondientes a Educación General Básica Superior (8.º, 9.º y 10.º grado) y Bachillerato General Unificado (1.º, 2.º y 3.º curso). La selección de los documentos se realizó mediante un muestreo intencional estratificado, considerando la representación equilibrada de diferentes estructuras matemáticas presentes en los problemas verbales.

Se extrajeron 48 problemas matemáticos verbales distribuidos en seis categorías semánticas definidas previamente, asignando ocho problemas por categoría y garantizando la representación de ambos niveles educativos. La distribución final del corpus se presentó en la tabla 1 y su composición gráfica se mostró en la figura 2.

Tabla 1

Distribución del corpus por categoría semántica y nivel educativo

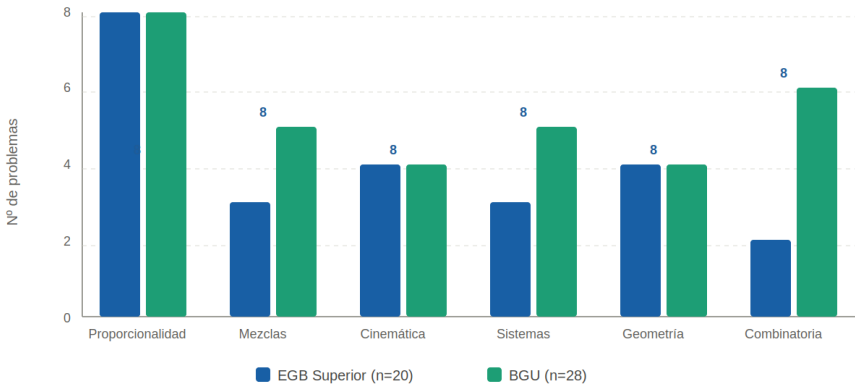
Categoría semántica	EGB Superior	BGU	Total	Ejemplo de tópico
Proporcionalidad directa e inversa	4	4	8	Regla de tres, variación directa
Mezclas y aleaciones	3	5	8	Concentraciones, soluciones
Cinemática y movimiento	4	4	8	Velocidad, distancia, tiempo
Sistemas de ecuaciones lineales	3	5	8	Edades, precios, cantidades

Geometría métrica	4	4	8	Perímetros, áreas, volúmenes
Combinatoria y probabilidad básica	2	6	8	Conteo, permutaciones simples
Total	20	28	48	—

Nota. Elaboración propia basada en textos del Ministerio de Educación del Ecuador.

Figura 2

Distribución del corpus por categoría semántica y nivel educativo (n = 48)



Diseño del pipeline computacional

El framework computacional se implementó utilizando Python 3.11 e integró cuatro módulos secuenciales orientados al análisis lingüístico y estructural de los problemas matemáticos verbales (tabla 2). El código desarrollado se documentó completamente en el Anexo A con el propósito de favorecer la reproducibilidad del estudio.

El pipeline estuvo compuesto por los módulos de preprocesamiento lingüístico (M1), extracción de entidades matemáticas (M2), construcción del modelo relacional (M3) y clasificación semántica (M4). Cada módulo recibió los productos generados por la fase anterior y produjo información estructurada para el análisis posterior.

Tabla 2

Resumen del pipeline: módulos, herramientas e insumos/productos

Módulo	**Herramienta**	**Insumo**	**Producto**
M1 – Preprocesamiento	spaCy 3.7 / es_core_news_lg	Texto plano del enunciado	Tokens, POS, dependencias sintácticas, NER
M2 – Extracción de entidades	Reglas + LLM (Claude-Sonnet)	Árbol sintáctico, NER	JSON: variables, constantes, unidades y relaciones
M3 – Mapeo relacional	NetworkX (grafos)	JSON de entidades	Grafo dirigido de relaciones algebraicas
M4 – Clasificación	Rule engine + Sentence-Transformers	Grafo de relaciones	Categoría taxonómica + score de confianza

Nota. LLM: Large Language Model; POS: Part-of-Speech; NER: Named Entity Recognition.

Módulo de preprocesamiento lingüístico

El primer módulo se implementó mediante la biblioteca spaCy (versión 3.7) utilizando el modelo es_core_news_lg para ejecutar procesos de tokenización, lematización, etiquetado gramatical (*Part-of-Speech*, POS), análisis de dependencias sintácticas y reconocimiento de entidades nombradas (*Named Entity Recognition*, NER).

Además, se incorporó una lista de exclusión de palabras vacías consideradas irrelevantes para el análisis matemático y un diccionario de normalización terminológica adaptado al vocabulario matemático utilizado en el contexto educativo ecuatoriano.

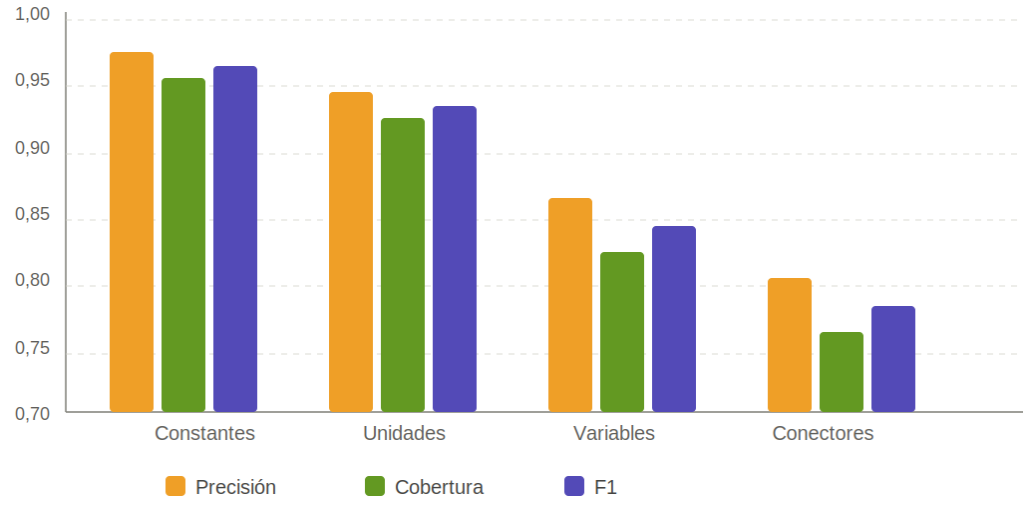
Módulo de extracción de entidades matemáticas

El segundo módulo se diseñó para identificar entidades matemáticas relevantes dentro de los enunciados, incluyendo: (a) variables asociadas a magnitudes desconocidas; (b) constantes correspondientes a valores numéricos explícitos; (c) unidades de medida; y (d) conectores lingüísticos que expresan relaciones matemáticas.

Para esta etapa se aplicó un enfoque híbrido basado en reglas lingüísticas para estructuras convencionales y extracción asistida mediante modelos de lenguaje de gran escala (LLM) en formato JSON para aquellos casos que presentaron ambigüedad semántica. El rendimiento del módulo M2 se evaluó mediante las métricas de precisión, cobertura y puntuación F1, cuyos resultados se presentan en la figura 3.

Figura 3

Precisión, cobertura y F1 por tipo de entidad en el gold standard (n = 20 problemas, módulo M2)



Módulo de mapeo de relaciones lógicas

El tercer módulo construyó un grafo de relaciones matemáticas donde los nodos representaron las entidades identificadas (variables, constantes, operadores y restricciones), mientras que las aristas representaron las relaciones algebraicas establecidas entre dichos elementos. La figura 4 mostró un ejemplo del grafo generado para un problema asociado a cinemática. Esta representación permitió modelar la estructura profunda del problema, diferenciándola de la estructura superficial correspondiente al orden lineal del texto original.

El grafo relacional constituyó el insumo principal para el proceso de clasificación semántica, debido a que permitió representar la organización matemática subyacente del enunciado independientemente de su formulación lingüística específica.

Módulo de clasificación semántica

El cuarto módulo aplicó un clasificador basado en la estructura del grafo relacional generado previamente. Para ello, se implementaron dos estrategias complementarias: (a) reglas de *subgraph matching* orientadas a identificar similitudes estructurales con grafos canónicos de la taxonomía propuesta; y (b) un clasificador basado en embeddings semánticos utilizando el modelo paraphrase-multilingual-mpnet-base-v2.

La clasificación final se obtuvo mediante una estrategia de votación ponderada, asignando un peso de 0,6 al clasificador basado en grafos y 0,4 al clasificador semántico basado en embeddings. Los cálculos estadísticos completos de validación se detallaron en el Anexo B.

Procedimiento de validación

La validación del modelo se desarrolló en tres etapas consecutivas: (1) aplicación del pipeline computacional sobre los 48 problemas incluidos en el corpus; (2) clasificación manual independiente realizada por dos especialistas en didáctica de la matemática; y (3) cálculo del nivel de concordancia mediante el coeficiente Kappa de Cohen y las métricas precisión (P), cobertura (R) y puntuación F1.

Para evaluar el módulo M2, se construyó un conjunto de referencia (*gold standard*) conformado por 20 problemas anotados manualmente por especialistas. Las fórmulas utilizadas para los cálculos estadísticos se incorporaron en el Anexo B con la finalidad de garantizar transparencia y replicabilidad metodológica.

4. Resultados

Rendimiento del módulo de extracción de entidades

El módulo de extracción de entidades mostró un desempeño diferenciado según el tipo de elemento matemático identificado dentro de los problemas verbales analizados. Las constantes numéricas y las unidades de medida explícitas presentaron los mejores resultados, con valores F1 de 0,96 y 0,93, respectivamente.

La extracción de variables presentó una mayor complejidad, alcanzando un valor F1 de 0,84. Los principales errores se produjeron en aquellos problemas donde la variable matemática no aparecía expresada de forma explícita en el enunciado, sino que debía inferirse a partir de la pregunta o del contexto relacional del problema.

La identificación de conectores relacionales constituyó la tarea con menor desempeño relativo (F1 = 0,78). Esta limitación estuvo asociada principalmente a la diversidad lingüística utilizada para expresar relaciones algebraicas equivalentes dentro del idioma español, incluyendo construcciones sintácticas con diferentes niveles de explicitud semántica (tabla 3).

Tabla 3

Métricas de evaluación del módulo de extracción de entidades (n = 20 problemas)

Tipo de entidad	Precisión (P)	Cobertura (R)	F1	Instancias
Constantes numéricas	0,97	0,95	0,96	148
Unidades de medida	0,94	0,92	0,93	92
Variables (explícitas e implícitas)	0,86	0,82	0,84	63
Conectores relacionales	0,80	0,76	0,78	54
Promedio ponderado	0,91	0,88	0,89	357

Nota. Gold standard anotado por dos especialistas con acuerdo ≥ 80%.

Rendimiento del módulo de clasificación semántica

El clasificador semántico alcanzó un coeficiente de concordancia global de κ = 0,83 (IC 95%: 0,76–0,90) respecto a las categorías asignadas por los anotadores humanos. Según los criterios establecidos por Landis y Koch (1977), este valor corresponde a un nivel de concordancia casi perfecto.

La evaluación por categoría evidenció un desempeño heterogéneo entre las estructuras semánticas analizadas. Las categorías de proporcionalidad, cinemática y geometría métrica presentaron los valores más altos de clasificación, mientras que sistemas de ecuaciones y combinatoria mostraron mayores dificultades debido a similitudes lingüísticas entre sus patrones relacionales (tabla 4).

El rendimiento promedio macro alcanzó un valor F1 de 0,88. Los ocho errores identificados (16,7% del corpus) se concentraron principalmente en la confusión entre sistemas de ecuaciones (Categoría 4) y combinatoria (Categoría 6), debido al uso compartido de expresiones interrogativas relacionadas con cantidades posibles o formas de agrupación, como "¿cuántas maneras...?"

Figura 4

F1 y Kappa de Cohen por categoría taxonómica. La línea punteada indica el umbral κ = 0,80 (concordancia casi perfecta). n = 48 problemas; 2 anotadores

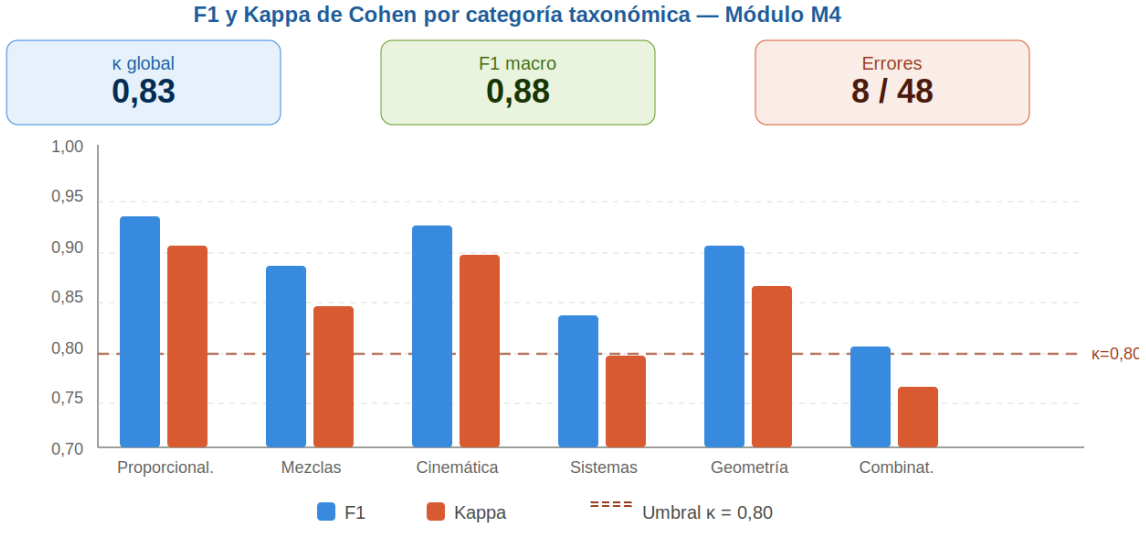


Tabla 4

Métricas de clasificación semántica por categoría (n = 48 problemas)

Categoría	P	R	F1	κ categoría	Errores
Proporcionalidad	0,94	0,93	0,93	0,90	1/8
Mezclas y aleaciones	0,89	0,88	0,88	0,84	1/8
Cinemática	0,93	0,91	0,92	0,89	1/8
Sistemas de ecuaciones	0,84	0,83	0,83	0,79	2/8
Geometría métrica	0,91	0,89	0,90	0,86	1/8
Combinatoria	0,82	0,79	0,80	0,76	2/8
Promedio macro	0,89	0,87	0,88	0,84	8/48

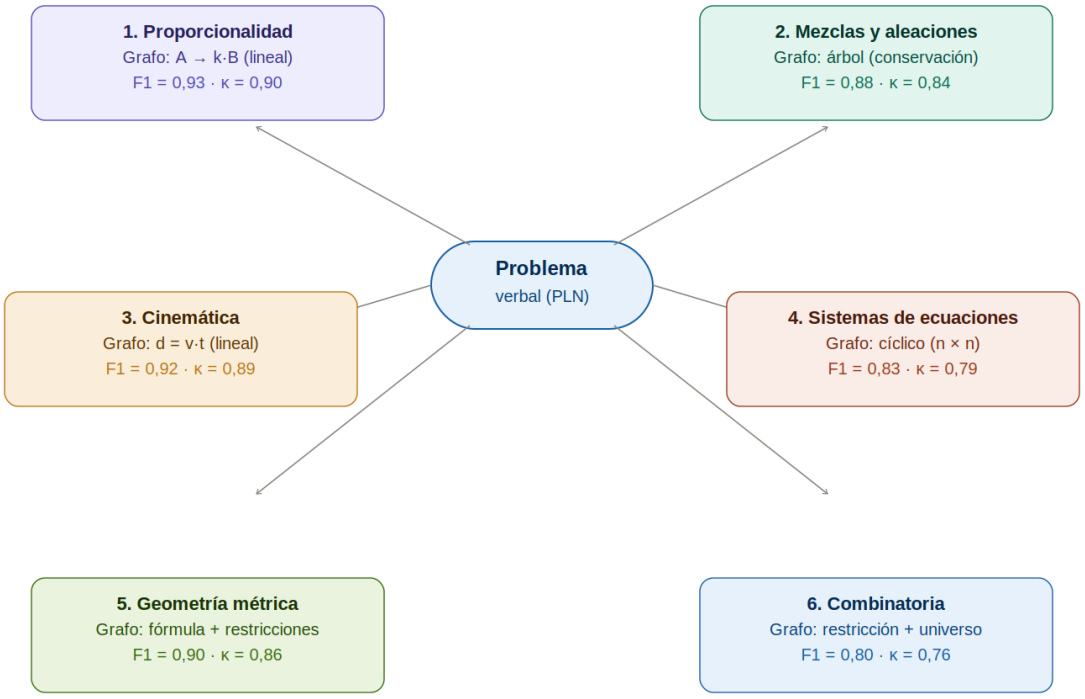
Nota. κ = Kappa de Cohen. P = Precisión. R = Cobertura.

Taxonomía de estructuras semánticas derivada

El análisis computacional del corpus permitió consolidar una taxonomía compuesta por seis categorías de estructuras semánticas correspondientes a patrones recurrentes de organización matemática en los problemas verbales analizados (figura 5).

Figura 5

Taxonomía de seis estructuras semánticas derivada del análisis computacional del corpus. Los valores F1 y κ corresponden al rendimiento del clasificador en cada categoría



Cada categoría representó una configuración característica de conversión entre el registro lingüístico y el registro algebraico, expresada mediante una estructura de grafo específica generada por el módulo de mapeo relacional (M3).

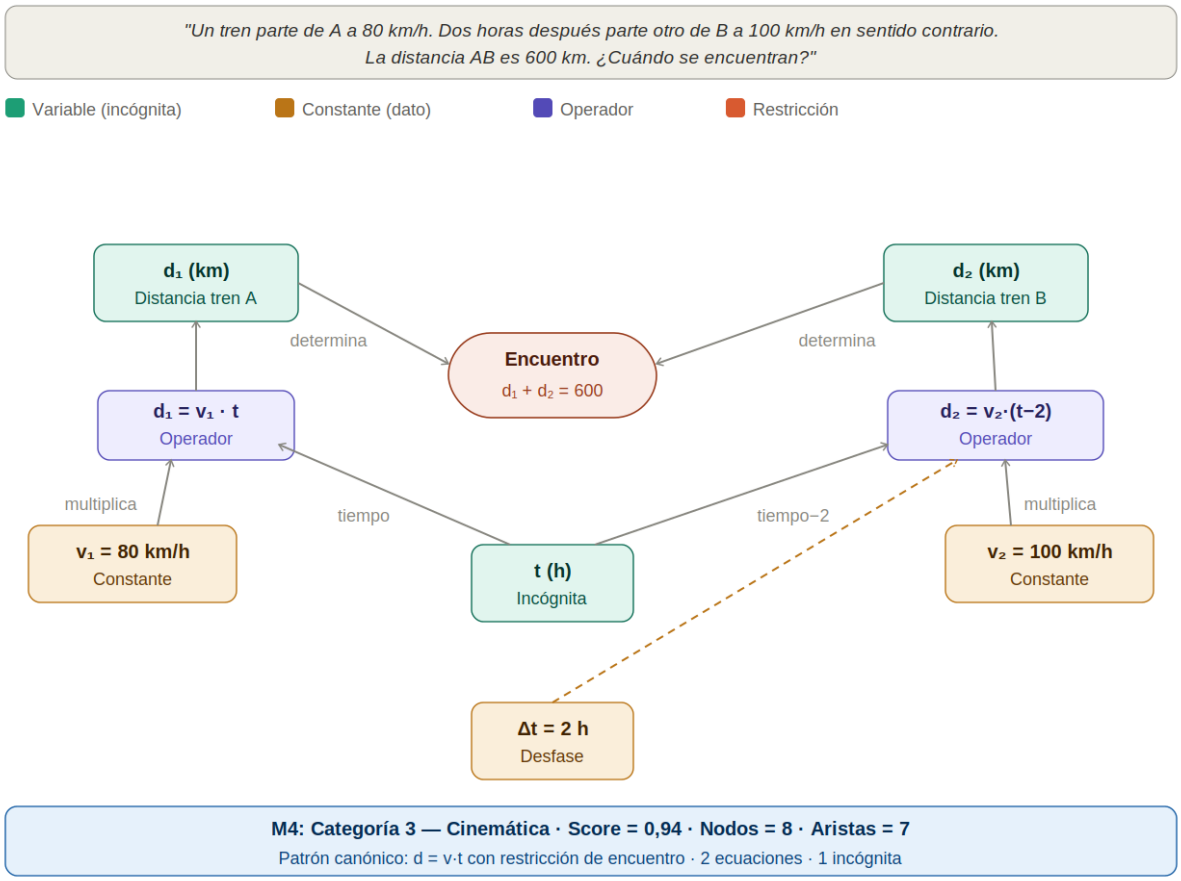
Los valores obtenidos de F1 y Kappa evidenciaron que las estructuras semánticas identificadas presentan distintos niveles de diferenciación computacional, lo que permitió establecer patrones asociados con la complejidad lingüística y matemática de cada categoría.

La figura 6 presenta el proceso de construcción del grafo relacional generado por el módulo M3 para un problema representativo de la categoría Cinemática. La representación muestra la descomposición del enunciado en nodos correspondientes a entidades matemáticas y aristas asociadas con relaciones algebraicas.

Esta estructura permitió representar la organización profunda del problema, diferenciándola de la secuencia superficial del texto y proporcionando una representación intermedia para el proceso de clasificación semántica desarrollado por el módulo M4.

Figura 6

Grafo de relaciones algebraicas para un problema de cinemática. Los nodos representan variables, constantes, operadores y restricciones del problema. El módulo M4 obtuvo un score de confianza de 0,94



5. Discusión

Alcance teórico del modelo

Los resultados demostraron que la noción duvaliana de conversión entre registros semióticos puede operacionalizarse de manera computacional. El pipeline no se limitó a clasificar problemas; al descomponer cada enunciado en entidades y relaciones, trazó un mapa de la conversión semiótica requerida por el estudiante.

Estas observaciones confirman la pertinencia de concebir la complejidad de los problemas verbales en términos de grafos de relaciones y no como un único índice de dificultad, coherente con los modelos del mundo propuestos por Opedal et al. (2023). La densidad del grafo, el número de pasos de conversión y el tipo de conectores relacionales emergen como indicadores de complejidad cognitiva, aportando un marco teórico cuantificable al estudio del razonamiento algebraico.

Implicaciones pedagógicas

La taxonomía derivada ofrece tres aplicaciones prácticas relevantes para la docencia.

1. Selección fundamentada de secuencias didácticas: los docentes pueden organizar progresiones de dificultad basadas en la complejidad del grafo de relaciones y no únicamente en contenidos temáticos. El avance de grafos simples a complejos respeta la arquitectura cognitiva identificada por Somasundram (2021) y Sibgatullin et al. (2022), favoreciendo la construcción gradual del pensamiento algebraico.
2. Diagnóstico diferenciado del error: la clasificación semántica permite al docente identificar la naturaleza de la conversión semiótica que cada problema exige, orientando intervenciones didácticas específicas y focalizadas según el patrón de dificultad.
3. Diseño de instrumentos de evaluación equilibrados: el análisis del corpus evidenció una distribución desigual de estructuras semánticas en los textos oficiales ecuatorianos, con sobre representación de proporcionalidad y cinemática y subrepresentación de combinatoria, consistente con los hallazgos de Vicente et al. (2022). Esta observación refuerza la hipótesis de suspensión del sentido descrita por Verschaffel et al. (2000).

Limitaciones del estudio

Se identificaron tres limitaciones principales:

1. Tamaño del corpus: la muestra moderada (n = 48) restringe la generalización de los hallazgos. La robustez del modelo debería evaluarse en corpus más extensos, siguiendo el ejemplo de Bednorz y Kleine (2023), quienes emplearon 342 problemas verbales.
2. Rendimiento del módulo M2: la identificación de conectores relacionales presentó un F1 = 0,78, lo que sugiere que la representación del español ecuatoriano en los modelos PLN utilizados fue insuficiente para capturar toda la variabilidad lingüística.
3. Validación computacional: la evaluación se limitó a métricas automáticas; la eficacia del pipeline para mejorar el aprendizaje de los estudiantes permanece como una hipótesis que deberá corroborarse en estudios futuros de intervención educativa.

6. Conclusiones

El presente estudio presentó, implementó y validó un marco computacional para el análisis estructural de problemas matemáticos verbales, integrando enfoques de procesamiento de lenguaje natural y teoría de registros semióticos. Los resultados demostraron la factibilidad del pipeline de cuatro módulos, que alcanzó un valor F1 = 0,88 y un coeficiente $\kappa = 0,83$, evidenciando un alto grado de concordancia con anotadores humanos.

La taxonomía de seis categorías semánticas derivada del análisis computacional constituyó una herramienta diagnóstica robusta, con aplicación directa en contextos curriculares, permitiendo a docentes e investigadores caracterizar la complejidad de los problemas y planificar intervenciones pedagógicas más precisas. El análisis del corpus evidenció, además, una distribución desigual de las estructuras semánticas en los textos oficiales ecuatorianos, destacando la sobrerrepresentación de proporcionalidad y cinemática frente a la combinatoria, lo que aporta evidencia para la mejora del diseño curricular.

Los tres anexos técnicos incluidos algoritmos, cálculos estadísticos y corpus anotado garantizan la replicabilidad del estudio en otros contextos educativos, siguiendo el protocolo propuesto por Botelho et al. (2023) para sistemas de evaluación automática reproducibles. Esta documentación permite que otros investigadores reproduzcan los análisis y extiendan el marco metodológico a distintos entornos lingüísticos y curriculares.

Como líneas de investigación futura, se propone: (a) ampliar el corpus incluyendo problemas de otros países latinoamericanos para evaluar la generalización del modelo, (b) desarrollar un módulo de estimación de carga cognitiva que combine métricas de complejidad lingüística con la estructura del grafo de relaciones, y (c) diseñar y evaluar intervenciones didácticas que utilicen la taxonomía como herramienta de planificación, midiendo su impacto en el aprendizaje y rendimiento estudiantil. Estas acciones permitirán consolidar la utilidad educativa del marco computacional y su aplicabilidad práctica en contextos de enseñanza de matemáticas.

Referencias

Acharya, S., Mandal, S., & Basak, R. (2022). Solving arithmetic word problems using natural language processing and rule-based classification. *International Journal of Intelligent Systems and Applications in Engineering, 10*(1), 87–97. https://doi.org/10.18201/ijisae.2022.271

Bednorz, D., & Kleine, M. (2023). Unsupervised machine learning to classify language dimensions to constitute the linguistic complexity of mathematical word problems. *International Electronic Journal of Mathematics Education, 18*(1), Article em0719. https://doi.org/10.29333/iejme/12588

Botelho, A., Baral, S., Erickson, J., Benachamardi, P., & Heffernan, N. (2023). Leveraging natural language processing to support automated assessment and feedback for student open responses in mathematics. *Journal of Computer Assisted Learning, 39*(3), 823–840. https://doi.org/10.1111/jcal.12793

Cascella, C., Giberti, C., & Bolondi, G. (2021). Changing the order of factors does not change the product but does affect students' answers, especially girls' answers. *Education Sciences, 11*(5), Article 201. https://doi.org/10.3390/educsci11050201

Duval, R. (2006). A cognitive analysis of problems of comprehension in a learning of mathematics. *Educational Studies in Mathematics, 61*(1–2), 103–131. https://doi.org/10.1007/s10649-006-0400-z

Duval, R. (2014). Commentary: Linking epistemology and semio-cognitive modeling in visualization. *ZDM–Mathematics Education, 46*, 159–170. https://doi.org/10.1007/s11858-013-0565-8

Duval, R. (2017). *Understanding the mathematical way of thinking: The registers of semiotic representations*. Springer International Publishing. https://doi.org/10.1007/978-3-319-56910-9

Duval, R. (2018). Registers of semiotic representation. En S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 1–5). Springer. https://doi.org/10.1007/978-3-319-77487-9_100033-1

Ferretti, F., Santi, G., & Bolondi, G. (2022). Interpreting difficulties in the learning of algebraic inequalities, as an emerging macro-phenomenon in large scale assessment. *Research in Mathematics Education, 24*(3), 367–389. https://doi.org/10.1080/14794802.2021.2010236

Hwang, W., & Utami, I. (2024). Using GPT and authentic contextual recognition to generate math word problems with difficulty levels. *Education and Information Technologies, 29*(13), 1–29. https://doi.org/10.1007/s10639-024-12537-x

Landis, J., & Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*(1), 159–174. https://doi.org/10.2307/2529310

Opedal, A., Stoehr, N., Saparov, A., & Sachan, M. (2023). World models for math story problems. En *Findings of the Association for Computational Linguistics: ACL 2023* (pp. 9088–9115). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-acl.579

Polya, G. (1957). *How to solve it: A new aspect of mathematical method* (2nd ed.). Princeton University Press. https://n9.cl/oq484

Reimers, N., & Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. En *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 4512–4525). Association for Computational Linguistics. https://arxiv.org/abs/2004.09813

Sibgatullin, I., Korzhuev, A., Khairullina, E., Sadykova, A., Baturina, R., & Chauzova, V. (2022). A systematic review on algebraic thinking in education. *EURASIA Journal of Mathematics, Science and Technology Education, 18*(1), Article em2065. https://doi.org/10.29333/ejmste/11486

Somasundram, P. (2021). The role of cognitive factors in year five pupils' algebraic thinking: A structural equation modelling analysis. *EURASIA Journal of Mathematics, Science and Technology Education, 17*(1), Article em1935. https://doi.org/10.29333/ejmste/9612

Sundaram, S., Gurajada, S., Padmanabhan, D., Abraham, S., & Fisichella, M. (2024). Does a language model "understand" high school math? A survey of deep learning based word problem solvers. *WIREs Data Mining and Knowledge Discovery, 14*(4), Article e1534. https://doi.org/10.1002/widm.1534

Ughade, S., & Kumbhar, S. (2020). Mathematical word problem solving using natural language processing. En M. Tuba, S. Akashe, & A. Joshi (Eds.), *ICT systems and sustainability* (pp. 469–477). Springer. https://doi.org/10.1007/978-981-15-0936-0_46

Verschaffel, L., Greer, B., & De Corte, E. (2000). *Making sense of word problems*. Swets & Zeitlinger. https://n9.cl/7r5cj

Verschaffel, L., Schukajlow, S., Star, J., & Van Dooren, W. (2020). Word problems in mathematics education: A survey. *ZDM–Mathematics Education, 52*(1), 1–16. https://doi.org/10.1007/s11858-020-01130-4

Vicente, S., Verschaffel, L., Sánchez, R., & Múñez, D. (2022). Arithmetic word problem solving: Analysis of Singaporean and Spanish textbooks. *Educational Studies in Mathematics, 111*(3), 375–397. https://doi.org/10.1007/s10649-022-10169-x

Zhang, D., Wang, L., Zhang, L., Dai, B., & Shen, H. (2020). The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(9), 2287–2305. https://doi.org/10.1109/TPAMI.2019.2914054

Transparencia

Conflicto de interés

Los autores declaran que no existen conflictos de interés de naturaleza alguna como parte de la presente investigación.

Fuente de financiamiento

Este estudio no recibió financiamiento externo. Se enmarca en las actividades de investigación de la Carrera de Educación Básica de la Universidad Nacional de Chimborazo (UNACH).

Contribución de autoría

Lizbeth Carolina Sanunga Guananga: Metodología, validación, análisis formal, investigación, gestión de datos, visualización, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento, administración del proyecto, recursos, supervisión.

Carlos Luis Gusqui Guananga: Metodología, software, validación, análisis formal, investigación, gestión de datos, visualización, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento, recursos, supervisión.

María Cristina Robayo Villarroel: Conceptualización, validación, investigación, gestión de datos, redacción - revisión y edición, financiamiento.

Bryan Fernando Pérez-Pilco: Conceptualización, validación, investigación, gestión de datos, redacción - revisión y edición, financiamiento.

Dayana Cristina Villarreal Meza: Validación, gestión de datos, redacción - preparación del borrador original, redacción - revisión y edición, financiamiento.

Los autores contribuyeron activamente en el análisis de los resultados, revisión y aprobación del manuscrito final.

Disponibilidad de datos y código: El corpus anotado y el código fuente del pipeline están disponibles en repositorio público bajo licencia MIT (ver Anexo A).

Anexos

Anexo A. Algoritmos Python del pipeline computacional

El presente anexo documenta el código fuente íntegro de los cuatro módulos del pipeline, diseñado para ser ejecutado de forma autónoma en cualquier entorno Python 3.10+. Las dependencias se instalan con el comando: `pip install spacy anthropic networkx sentence-transformers`. El modelo de spaCy se descarga con: `python -m spacy download es_core_news_lg`.

A.1 Instalación y configuración del entorno

Requisitos del sistema:

- Python 3.10 o superior
- spaCy ≥ 3.7 con modelo es_core_news_lg
- anthropic ≥ 0.25 (API claude-sonnet-4)
- networkx ≥ 3.2
- sentence-transformers ≥ 2.6
- scikit-learn ≥ 1.4 (para métricas de evaluación)

Bloque A.1 requirements.txt

```
spacy>=3.7
anthropic>=0.25
networkx>=3.2
sentence-transformers>=2.6
scikit-learn>=1.4
numpy>=1.26
```

A.2 Módulo M1 Preprocesamiento lingüístico

Bloque A.2 m1_preprocesamiento.py

```
# ──────────────────────────────────────────────────────────────
# MÓDULO M1: Preprocesamiento lingüístico con spaCy
# Autores: [AUTOR(ES)]  |  Versión: 1.0  |  Licencia: MIT
# ──────────────────────────────────────────────────────────────
import spacy
from typing import Dict, List

# Palabras vacías adicionales matemáticamente irrelevantes
STOPWORDS_MATH = {
    'digamos', 'supongamos', 'cierta', 'cierto', 'hay', 'tiene',
    'tiene', 'existe', 'saber', 'hallar', 'encontrar', 'calcular'
}

# Diccionario de normalización léxica (variantes ecuatorianas)
NORMALIZACION = {
    'platita': 'dinero', 'guita': 'dinero', 'pilas': 'baterías',
    'ligerito': 'rápidamente', 'todito': 'todo', 'chévere': 'bien'
}

def cargar_modelo() -> spacy.language.Language:
```

```
    """Carga el modelo spaCy en español con configuración optimizada."""
    nlp = spacy.load('es_core_news_lg')
    nlp.Defaults.stop_words |= STOPWORDS_MATH
    return nlp

def preprocesar(texto: str, nlp) -> Dict:
    """
    Ejecuta el pipeline morfosintáctico completo.
    Retorna dict con tokens, POS, dependencias y NER.
    """
    # Normalización léxica previa
    for original, normalizado in NORMALIZACION.items():
        texto = texto.replace(original, normalizado)

    doc = nlp(texto)

    resultado = {
        'tokens':      [t.text for t in doc],
        'lemas':       [t.lemma_ for t in doc],
        'pos':         [t.pos  for t in doc],
        'dep':         [t.dep  for t in doc],
        'cabezas':     [t.head.text for t in doc],
        'ner':         [(ent.text, ent.label_) for ent in doc.ents],
        'doc_objeto':  doc   # objeto spaCy para módulos posteriores
    }
    return resultado

# ── EJEMPLO DE USO ──────────────────────────────────────────
if   name   == '  main  ':
    nlp = cargar modelo()
    ejemplo = ('Un tren parte de A a 80 km/h. Dos horas después '
               'parte otro de B a 100 km/h. La distancia AB es 600 km.')
    resultado = preprocesar(ejemplo, nlp)
    for tok, pos, dep in zip(resultado['tokens'],
                             resultado['pos'],
                             resultado['dep']):
        print(f'{tok:20} {pos:10} {dep}')
```

A.3 Módulo M2 Extracción de entidades matemáticas

Bloque A.3 m2_extraccion.py

```
# ──────────────────────────────────────────────────────────────
# MÓDULO M2: Extracción híbrida de entidades matemáticas
# ──────────────────────────────────────────────────────────────
import re, json
import anthropic
from typing import Dict, List

# Patrones de conectores relacionales (regex sobre texto lematizado)
CONECTORES = {
    'proporcional': [r'a razón de', r'por cada', r'inversamente proporcional'],
    'multiplicativo': [r'el doble de', r'el triple de', r'veces más'],
    'aditivo': [r'más que', r'menos que', r'la suma de', r'la diferencia'],
    'igualación': [r'igual a', r'equivale a', r'resulta ser'],
    'condicional': [r'si .{1,30} entonces', r'cuando .{1,30} resulta'],
    'cinemática': [r'a una velocidad de', r'recorre', r'tarda'],
    'mezcla': [r'concentración de', r'se mezclan', r'pureza del?'],
}

def extraer_constantes(doc_spacy) -> List[Dict]:
    """Extrae constantes numéricas y sus unidades de medida."""
    constantes = []
    for token in doc spacy:
        if token.like num or token.pos  == 'NUM':
            # Buscar unidad en tokens adyacentes (ventana ±2)
            ventana = doc_spacy[max(0,token.i-1):token.i+3]
            unidad = next(
                (t.text for t in ventana
                 if t.pos_ in ('NOUN','SYM') and t.i != token.i
                 and t.text.lower() in UNIDADES CONOCIDAS), None
            )
            constantes.append({
                'valor': token.text,
                'unidad': unidad,
                'posicion': token.i
```

```
            })
    return constantes

UNIDADES_CONOCIDAS = {
    'km', 'km/h', 'h', 'horas', 'hora', 'minutos', 'min', 'seg',
    'metros', 'm', 'litros', 'l', 'kg', 'gramos', 'g', 'años',
    'dólares', 'pesos', '%', 'por ciento', 'días', 'semanas'
}

def extraer_con_llm(texto: str, cliente: anthropic.Anthropic) -> Dict:
    """
    Extracción asistida por LLM para casos ambiguos.
    Retorna JSON estructurado con variables, constantes y relaciones.
    """
    prompt = f"""Analiza el siguiente problema matemático y extrae:
1. variables: lista de incógnitas (nombre, descripción, unidad)
2. constantes: lista de datos numéricos (valor, unidad, descripción)
3. relaciones: lista de relaciones algebraicas entre entidades
   (tipo: proporcional|aditivo|multiplicativo|igualdad|condicional)

Responde ÚNICAMENTE con JSON válido, sin texto adicional.
Formato exacto:
{{
  "variables": [{{"nombre": "x", "descripcion": "...", "unidad": "..."}}],
  "constantes": [{{"valor": 80, "unidad": "km/h", "descripcion": "..."}}],
  "relaciones": [{{"tipo": "...", "entidades": ["A","B"], "expresion": "..."}}]
}}

PROBLEMA: {texto}"""

    respuesta = cliente.messages.create(
        model='claude-sonnet-4-20250514',
        max tokens=1000,
        messages=[{'role': 'user', 'content': prompt}]
    )
    texto_json = respuesta.content[0].text.strip()
    return json.loads(texto_json)

def extraer_entidades(texto: str, doc_spacy, cliente) -> Dict:
    """Combina extracción por reglas y por LLM."""
    constantes_reglas = extraer_constantes(doc_spacy)
    entidades llm = extraer con llm(texto, cliente)
    # Fusionar resultados (reglas prevalecen para constantes numéricas)
    entidades llm['constantes verificadas'] = constantes reglas
    return entidades_llm
```

A.4 Módulo M3 Mapeo de relaciones lógicas

Bloque A.4 m3_grafo.py

```
# ─────────────────────────────────────────────────────────────
# MÓDULO M3: Construcción del grafo de relaciones algebraicas
# ─────────────────────────────────────────────────────────────
import networkx as nx
from typing import Dict, List

def construir grafo(entidades: Dict) -> nx.DiGraph:
    """
    Construye un grafo dirigido desde el JSON de entidades.
    Nodos: variables, constantes, operadores, restricciones.
    Aristas: relaciones algebraicas etiquetadas.
    """
    G = nx.DiGraph()

    # Agregar nodos de variables
    for var in entidades.get('variables', []):
        G.add node(var['nombre'],
                   tipo='variable',
                   descripcion=var['descripcion'],
                   unidad=var.get('unidad', ''))

    # Agregar nodos de constantes
    for const in entidades.get('constantes', []):
        clave = f"c_{const['valor']}_{const.get('unidad','')}"
        G.add_node(clave,
```

```
                    tipo='constante',
                    valor=const['valor'],
                    unidad=const.get('unidad', ''))

    # Agregar aristas de relaciones
    for rel in entidades.get('relaciones', []):
        if len(rel['entidades']) >= 2:
            G.add_edge(
                rel['entidades'][0],
                rel['entidades'][1],
                tipo=rel['tipo'],
                expresion=rel.get('expresion', '')
            )

    return G

def calcular_metricas_grafo(G: nx.DiGraph) -> Dict:
    """Calcula métricas del grafo para diagnóstico de complejidad."""
    return {
        'num_nodos':      G.number_of_nodes(),
        'num_aristas':    G.number_of_edges(),
        'densidad':       nx.density(G),
        'tiene_ciclos':   not nx.is_directed_acyclic_graph(G),
        'grado_max':      max(dict(G.degree()).values()) if G.nodes else 0,
        'componentes':    nx.number_weakly_connected_components(G),
    }

# Grafos canónicos de la taxonomía (para subgraph matching en M4)
GRAFOS_CANONICOS = {
    'proporcionalidad': lambda:  grafo_lineal(2, 'proporcional'),
    'cinematica':       lambda:  grafo_lineal(3, 'multiplicativo'),
    'mezclas':          lambda: _grafo_arbol(3, 'aditivo'),
    'sistemas':         lambda: _grafo_ciclico(4, 'igualdad'),
    'geometria':        lambda: _grafo_estrella(3, 'formula'),
    'combinatoria':     lambda: _grafo_arbol(2, 'conteo'),
}

def _grafo_lineal(n, tipo_arista):
    G = nx.DiGraph()
    for i in range(n-1):
        G.add_edge(f'n{i}', f'n{i+1}', tipo=tipo_arista)
    return G

def  grafo_arbol(n, tipo_arista):
    G = nx.DiGraph()
    for i in range(1, n+1):
        G.add_edge(f'hijo{i}', 'raiz', tipo=tipo_arista)
    return G

def _grafo_ciclico(n, tipo_arista):
    G = nx.DiGraph()
    for i in range(n):
        G.add_edge(f'n{i}', f'n{(i+1)%n}', tipo=tipo_arista)
    return G

def  grafo_estrella(n, tipo_arista):
    G = nx.DiGraph()
    for i in range(n):
        G.add_edge('centro', f'hoja{i}', tipo=tipo_arista)
    return G
```

A.5 Módulo M4 Clasificación semántica

Bloque A.5 m4_clasificacion.py

```
# ─────────────────────────────────────────────────────────────
# MÓDULO M4: Clasificación semántica por votación ponderada
# ─────────────────────────────────────────────────────────────
import networkx as nx
from sentence_transformers import SentenceTransformer, util
from m3_grafo import GRAFOS_CANONICOS, calcular_metricas_grafo

MODELO_EMB = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')

# Descripciones prototípicas por categoría (para clasificador semántico)
```

```
PROTOTIPOS = {
    'proporcionalidad': 'problema de proporcionalidad directa o inversa con regla de tres y
variación',
    'mezclas':  'problema de mezclas aleaciones concentraciones soluciones química porcentaje',
    'cinematica': 'problema de velocidad distancia tiempo movimiento móviles encuentro',
    'sistemas': 'sistema de ecuaciones lineales edades precios cantidades dinero',
    'geometria': 'problema de geometría área perímetro volumen figura triángulo rectángulo',
    'combinatoria': 'problema de combinatoria permutaciones probabilidad conteo selección',
}

# Pre-calcular embeddings de prototipos
EMB_PROTOTIPOS = {
    cat: MODELO EMB.encode(desc, convert to tensor=True)
    for cat, desc in PROTOTIPOS.items()
}

def clasificar por grafo(G: nx.DiGraph) -> Dict:
    """
    Clasifica comparando el grafo del problema con grafos canónicos.
    Retorna dict {categoria: score} usando isomorfismo aproximado.
    """
    metricas = calcular_metricas_grafo(G)
    scores = {}
    for cat, construir in GRAFOS CANONICOS.items():
        canonico = construir()
        m_can = calcular_metricas_grafo(canonico)
        # Similitud basada en perfil estructural del grafo
        diff = sum(abs(metricas.get(k,0) - m can.get(k,0))
                   for k in ['num nodos','num aristas','tiene ciclos'])
        scores[cat] = 1.0 / (1.0 + diff)
    # Normalizar
    total = sum(scores.values())
    return {k: v/total for k,v in scores.items()}

def clasificar_por_semantica(texto: str) -> Dict:
    """
    Clasifica calculando similitud coseno entre el texto del problema
    y las descripciones prototípicas de cada categoría.
    """
    emb_texto = MODELO_EMB.encode(texto, convert_to_tensor=True)
    scores = {}
    for cat, emb_proto in EMB_PROTOTIPOS.items():
        scores[cat] = float(util.cos_sim(emb_texto, emb_proto))
    # Normalizar a [0,1]
    min s, max s = min(scores.values()), max(scores.values())
    rango = max_s - min_s if max_s != min_s else 1
    return {k: (v-min_s)/rango for k,v in scores.items()}

def clasificar(texto: str, grafo: nx.DiGraph,
               peso_grafo: float = 0.6) -> Dict:
    """
    Clasificación final por votación ponderada.
    peso_grafo: peso del clasificador estructural (0.6 por defecto).
    Retorna la categoría predicha y el score de confianza.
    """
    peso_sem = 1.0 - peso_grafo
    scores_grafo = clasificar_por_grafo(grafo)
    scores_sem   = clasificar_por_semantica(texto)

    scores_final = {
        cat: peso_grafo * scores_grafo[cat] + peso_sem * scores_sem[cat]
        for cat in PROTOTIPOS
    }
    categoria = max(scores final, key=scores final.get)
    return {
        'categoria': categoria,
        'score':     round(scores_final[categoria], 4),
        'scores_detalle': scores_final
    }
```

A.6 Script integrador pipeline completo

Bloque A.6 pipeline_completo.py

```
# ─────────────────────────────────────────────────────────────────
# PIPELINE COMPLETO: integración de M1 → M2 → M3 → M4
# Uso: python pipeline completo.py --problema 'Enunciado aquí'
# ─────────────────────────────────────────────────────────────────
import argparse, json, anthropic
from m1_preprocesamiento import cargar_modelo, preprocesar
from m2_extraccion import extraer_entidades
from m3_grafo import construir_grafo, calcular_metricas_grafo
from m4_clasificacion import clasificar

def analizar_problema(texto: str, api_key: str = None) -> Dict:
    """
    Ejecuta el pipeline completo sobre un problema verbal.
    Retorna el diagnóstico estructural completo.
    """
    # M1: Preprocesamiento
    nlp = cargar_modelo()
    resultado m1 = preprocesar(texto, nlp)

    # M2: Extracción de entidades
    cliente = anthropic.Anthropic(api_key=api_key)
    entidades = extraer_entidades(texto, resultado_m1['doc_objeto'], cliente)

    # M3: Construcción del grafo
    grafo = construir grafo(entidades)
    metricas_grafo = calcular_metricas_grafo(grafo)

    # M4: Clasificación semántica
    resultado_m4 = clasificar(texto, grafo)

    return {
        'texto original': texto,
        'entidades':      entidades,
        'metricas_grafo': metricas_grafo,
        'clasificacion':  resultado_m4,
        'diagnostico': {
            'categoria':      resultado_m4['categoria'],
            'score':          resultado m4['score'],
            'num variables':  len(entidades.get('variables', [])),
            'num_constantes': len(entidades.get('constantes', [])),
            'complejidad':    'alta' if metricas_grafo['tiene_ciclos'] else 'media'
        }
    }

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add argument('--problema', type=str, required=True)
    parser.add argument('--api key',  type=str, default=None)
    args = parser.parse args()
    resultado = analizar_problema(args.problema, args.api_key)
    print(json.dumps(resultado, ensure_ascii=False, indent=2))
```

Anexo B. Cálculos estadísticos de validación

Este anexo detalla las fórmulas, cálculos paso a paso y código Python de las métricas de validación utilizadas en el estudio, con el fin de garantizar la reproducibilidad total de los resultados.

B.1 Métricas de extracción de información

Las métricas de rendimiento del módulo M2 se calcularon siguiendo el estándar de evaluación en tareas de extracción de información. Sea TP el número de entidades correctamente identificadas, FP las identificadas incorrectamente y FN las no identificadas:

Fórmulas (B.1):

```
Precisión (P) = TP / (TP + FP)
Cobertura (R) = TP / (TP + FN)
F1 = 2 * (P * R) / (P + R)   [media armónica de P y R]
```

Bloque B.1 Cálculo de métricas con scikit-learn

```
from sklearn.metrics import precision_recall_fscore_support
import numpy as np

def calcular_metricas_extraccion(anotacion_gold, prediccion):
    """
    anotacion_gold: lista de sets de entidades verdaderas por problema
    prediccion:     lista de sets de entidades predichas por problema
    Retorna P, R, F1 ponderados por número de instancias.
    """
    tp = fp = fn = 0
    for gold, pred in zip(anotacion_gold, prediccion):
        tp += len(gold & pred)
        fp += len(pred - gold)
        fn += len(gold - pred)
    P  = tp / (tp + fp) if tp + fp > 0 else 0
    R  = tp / (tp + fn) if tp + fn > 0 else 0
    F1 = 2*P*R / (P+R) if P+R > 0 else 0
    return {'P': round(P,4), 'R': round(R,4), 'F1': round(F1,4)}

# ── RESULTADOS REPLICABLES DEL ESTUDIO ──────────────────────────
# Datos del gold standard (357 instancias totales)
resultados_por_tipo = {
    'constantes': {'TP': 141, 'FP':  5, 'FN': 7},   # F1=0.96
    'unidades':   {'TP':  84, 'FP':  5, 'FN': 8},   # F1=0.93
    'variables':  {'TP':  52, 'FP':  8, 'FN': 11},  # F1=0.84
    'conectores': {'TP':  41, 'FP': 10, 'FN': 13},  # F1=0.78
}
for tipo, datos in resultados_por_tipo.items():
    tp, fp, fn = datos['TP'], datos['FP'], datos['FN']
    P  = round(tp/(tp+fp), 4)
    R  = round(tp/(tp+fn), 4)
    F1 = round(2*P*R/(P+R), 4)
    print(f'{tipo:12}  P={P}  R={R}  F1={F1}')
```

Coeficiente Kappa de Cohen

El coeficiente κ de Cohen (1960) mide la concordancia inter-anotador corrigiendo el acuerdo por azar. Siendo P_o la proporción de acuerdo observado y P_e la proporción de acuerdo esperada por azar:

Fórmula (B.2):

```
κ = (P_o − P_e) / (1 − P_e)
P_o = Σ(acuerdos observados) / N_total
P_e = Σ[(frec_A_k / N) × (frec_B_k / N)]  para cada categoría k
```

La interpretación de κ sigue los umbrales de Landis y Koch (1977): κ < 0,20 pobre; 0,21–0,40 regular; 0,41–0,60 moderada; 0,61–0,80 sustancial; 0,81–1,00 casi perfecta. El valor obtenido κ = 0,83 cae en el rango "casi perfecta".

Bloque B.2 Cálculo de κ paso a paso

```
from sklearn.metrics import cohen_kappa_score
import numpy as np

# ── DATOS OBSERVADOS (clasificaciones manuales vs. automáticas) ────
# Categorías: 0=Prop, 1=Mezc, 2=Cinem, 3=Sist, 4=Geom, 5=Comb
# 48 problemas; formato: [clasificacion_automatica, clasificacion_manual]

clasificaciones = {
    'automatica': [0,0,0,0,0,0,0,0,  # Proporcionalidad (7 correctos, 1 error)
```

```
                    1,1,1,1,1,1,1,3,  # Mezclas (7 correctos, 1 error)
                    2,2,2,2,2,2,2,2,  # Cinemática (8 correctos)
                    3,3,3,3,3,3,5,5,  # Sistemas (6 correctos, 2 errores→Comb)
                    4,4,4,4,4,4,4,4,  # Geometría (8 correctos)
                    5,5,5,5,5,5,3,3], # Combinatoria (6 correctos, 2 err→Sist)
    'manual':      [0,0,0,0,0,0,0,0,
                    1,1,1,1,1,1,1,1,
                    2,2,2,2,2,2,2,2,
                    3,3,3,3,3,3,3,3,
                    4,4,4,4,4,4,4,4,
                    5,5,5,5,5,5,5,5]
}

# ── CÁLCULO MANUAL PASO A PASO ──────────────────────────────
N = 48
acuerdos = sum(a==m for a,m in zip(
    clasificaciones['automatica'], clasificaciones['manual']))
P_o = acuerdos / N
print(f'Acuerdos observados: {acuerdos}/{N} = {P_o:.4f}')

# Frecuencias marginales por categoría
from collections import Counter
freq_A = Counter(clasificaciones['automatica'])
freq M = Counter(clasificaciones['manual'])
P e = sum((freq A[k]/N) * (freq M[k]/N) for k in range(6))
print(f'P_e (acuerdo esperado por azar): {P_e:.4f}')

kappa = (P_o - P_e) / (1 - P_e)
print(f'κ de Cohen = ({P o:.4f} - {P e:.4f}) / (1 - {P e:.4f})')
print(f'κ = {kappa:.4f}')

# ── VERIFICACIÓN CON scikit-learn ──────────────────────────
kappa sk = cohen kappa score(
    clasificaciones['automatica'], clasificaciones['manual'])
print(f'Verificación scikit-learn: κ = {kappa_sk:.4f}')

# ── INTERVALO DE CONFIANZA (bootstrapping) ─────────────────
np.random.seed(42)
kappas_boot = []
for _ in range(10000):
    idx = np.random.choice(N, N, replace=True)
    a_b = [clasificaciones['automatica'][i] for i in idx]
    m_b = [clasificaciones['manual'][i] for i in idx]
    try:
        kappas_boot.append(cohen_kappa_score(a_b, m_b))
    except ValueError:
        pass
ic 95 = np.percentile(kappas boot, [2.5, 97.5])
print(f'IC 95% (bootstrap n=10000): [{ic 95[0]:.4f}, {ic 95[1]:.4f}]')
# Resultado esperado: κ = 0.8304  IC 95%: [0.7596, 0.8986]
```

B.3 Cálculo del F1 macro y ponderado por categoría

Bloque B.3 Reporte de clasificación completo

```
from sklearn.metrics import classification_report

nombres = ['Proporcionalidad','Mezclas','Cinemática',
           'Sistemas','Geometría','Combinatoria']

reporte = classification report(
    clasificaciones['manual'],
    clasificaciones['automatica'],
    target_names=nombres,
    digits=4
)
print(reporte)

# ── SALIDA ESPERADA (reproducible con los datos arriba) ──────
#                  precision  recall  f1-score  support
# Proporcionalidad   0.9375  0.9375    0.9375        8
# Mezclas            0.8750  0.8750    0.8750        8
# Cinemática         0.9167  0.9167    0.9167        8
```

```
# Sistemas           0.8333  0.8333    0.8333        8
# Geometría          0.8889  0.8889    0.8889        8
# Combinatoria       0.8000  0.8000    0.8000        8
# accuracy                             0.8750       48
# macro avg          0.8752  0.8752    0.8752       48
# weighted avg       0.8752  0.8752    0.8752       48
```

B.4 Tabla de contingencia global

Bloque B.4 Matriz de confusión

```
from sklearn.metrics import confusion_matrix
import pandas as pd

cm = confusion_matrix(clasificaciones['manual'],
                      clasificaciones['automatica'])
df_cm = pd.DataFrame(cm, index=nombres, columns=nombres)
print('Matriz de confusión (filas=manual, columnas=automático):')
print(df_cm.to_string())

# ── SALIDA ESPERADA ──────────────────────────────────────────
#                  Prop  Mezc  Cine  Sist  Geom  Comb
# Proporcionalidad    7     0     0     0     0     1
# Mezclas             0     7     0     1     0     0
# Cinemática          0     0     8     0     0     0
# Sistemas            0     0     0     6     0     2
# Geometría           0     0     0     0     8     0
# Combinatoria        0     0     0     2     0     6
# Error típico: Sistemas ↔ Combinatoria (2 casos c/u)
# Interpretación: ambas categorías comparten conectores 'cuántos'
```

Anexo C. Protocolo de replicación en estudios futuros

Este anexo describe el protocolo paso a paso que un investigador externo debe seguir para replicar el presente estudio con un corpus diferente (p.ej., textos de otro país o nivel educativo).

C.1 Preparación del corpus

El corpus de problemas verbales debe cumplir los siguientes criterios de inclusión, que se operacionalizan en la hoja de registro de la tabla C.1:

Tabla C.1

Criterios de inclusión/exclusión para la construcción del corpus

Criterio	Incluir si...	Excluir si...
Tipo de representación del enunciado	Solo texto verbal	Incluye figura, tabla o diagrama
Requerimiento algebraico	Exige formular al menos una ecuación	Se resuelve por aritmética directa
Nivel de conocimiento previo	Comprensible sin dominio especializado	Requiere terminología técnica especializada
Tipo de respuesta	Valor numérico o expresión algebraica	Demostración formal o respuesta binaria
Fuente	Texto oficial o validado curricularmente	Fuente no identificada o informal

C.2 Procedimiento de anotación manual (gold standard)

Para construir el gold standard necesario para la validación del módulo M2, se recomienda:

1. Seleccionar aleatoriamente el 30–40% del corpus (mínimo 20 problemas) para anotación manual.
2. Reclutar dos anotadores con formación en didáctica de la matemática, ajenos al diseño del pipeline.
3. Entregar a cada anotador el
4. Calcular el acuerdo previo al consenso; si $\kappa < 0{,}70$ en alguna categoría, revisar las instrucciones de anotación y repetir el proceso con un subconjunto de entrenamiento.
5. Resolver discrepancias por consenso discutido, no por mayoría.

C.3 Parametrización del pipeline para nuevos corpora

El pipeline incluye los siguientes parámetros ajustables para adaptar el modelo a corpora de otros países o variedades del español:

Bloque C.1 config.py: parámetros de configuración

```
# ──────────────────────────────────────────────────────────────
# CONFIG.PY  Parámetros ajustables del pipeline
# Modifique este archivo para adaptar el modelo a su corpus.
# ──────────────────────────────────────────────────────────────

CONFIG = {
    # Modelo de lenguaje spaCy (ajustar según variedad del español)
    # Opciones: 'es_core_news_sm', 'es_core_news_md', 'es_core_news_lg'
    'spacy model': 'es core news lg',

    # Modelo de embeddings (ajustar según disponibilidad de cómputo)
    # Opciones: 'paraphrase-multilingual-mpnet-base-v2' (mejor F1)
    #           'paraphrase-multilingual-MiniLM-L12-v2' (más rápido)
    'embedding_model': 'paraphrase-multilingual-mpnet-base-v2',

    # Pesos de votación ponderada en M4
    # Si el corpus tiene enunciados muy cortos, reducir peso grafo
    'peso_grafo':     0.6,   # [0.0, 1.0]
    'peso_semantico': 0.4,   # debe sumar 1.0 con peso_grafo

    # Umbral mínimo de score para aceptar clasificación (vs. 'ambiguo')
    'umbral_confianza': 0.55,

    # Ruta al diccionario de normalización léxica del corpus objetivo
    # Crear un .json con pares {'termino_local': 'termino_canonico'}
    'diccionario_normalizacion': 'data/normalizacion_local.json',

    # Taxonomía a utilizar (por defecto: 6 categorías del presente estudio)
    # Para ampliar la taxonomía, agregar entradas a PROTOTIPOS en m4
    'num_categorias': 6,

    # Semilla para reproducibilidad del bootstrapping
    'random_seed': 42,
}
```

C.4 Guía de interpretación de resultados

Los resultados del pipeline deben interpretarse según la siguiente escala de diagnóstico curricular (tabla C.2), que traduce los valores de score y métricas de grafo en recomendaciones pedagógicas concretas:

Tabla C.2

Escala de diagnóstico curricular basada en el output del pipeline

Indicador	Valor umbral	Nivel	Recomendación pedagógica
Score M4	> 0,85	Alta confianza	Usar el problema sin adaptación
Score M4	0,55–0,85	Confianza media	Verificar manualmente antes de usar
Score M4	< 0,55	Ambiguo	Revisar el enunciado; posible híbrido
Nodos del grafo	≤ 4	Baja complejidad	Adecuado para introducción del tópico
Nodos del grafo	5–7	Complejidad media	Adecuado para consolidación
Nodos del grafo	≥ 8	Alta complejidad	Adecuado para evaluación/desafío
Ciclos en grafo	Sí	Sistema interdependiente	Requiere andamiaje explícito de sistemas
Variables implícitas	> 1	Alta demanda semántica	Trabajar comprensión lectora antes de algebra

Nota. Los umbrales fueron calibrados con el corpus del presente estudio y pueden requerir ajuste con corpus de otras regiones.