



Universidad Veracruzana
Instituto de Investigaciones en Inteligencia Artificial

Detección de Ironía con NEAT y Análisis de Características

Presenta:

Lic. Daniela Juárez Morales

Director de tesis:

Dr. Guillermo de Jesús Hoyos Rivera

Co-directora de tesis:

Dra. Delia Irazú Hernández Farías

Instituto Nacional de Astrofísica, Óptica y Electrónica

Xalapa, Veracruz, México

Diciembre 2025



Agradecimientos

En primer lugar deseo expresar mi agradecimiento a mis directores de tesis, Dr. Guillermo de Jesús Hoyos Rivera, sus enseñanzas no solo enriquecieron este trabajo, sino que también dejaron una huella profunda en mi formación académica y personal. Gracias por confiar nuevamente en mí, por sus consejos siempre oportunos, por hablarme con firmeza cuando fue necesario y, sobre todo, por ser un verdadero mentor. A la Dra. Delia Irazú Hernández Farías, por su acompañamiento cercano, por abrirme las puertas del INAOE y por compartir conmigo sus conocimientos y la inspiración que me llevó a elegir este tema en particular. Gracias a ambos por confiar en mí y por la paciencia en los momentos en que más la necesité.

A mis padres, que han sido y seguirán siendo mi mayor fuente de motivación. Cada logro alcanzado es también suyo, fruto de su amor y de su apoyo incondicional. Son los mejores padres.

A la Universidad Veracruzana y al Instituto de Investigaciones en Inteligencia Artificial, por darme un espacio donde pude crecer como estudiante y como persona. A mis profesores, que compartieron conmigo sus conocimientos, la pasión por la investigación, y por sembrar en mí la motivación de siempre ir más allá.

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), su respaldo hizo posible que pudiera enfocarme plenamente en mi formación.

Y, por supuesto, a mis amigos: a los de toda la vida y a los que encontré en el instituto. Al equipo de Syntagma, Clemente y Juan, gracias por ser un apoyo indispensable en esta etapa, por resolver mis dudas en la madrugada y por siempre ayudarme. Sus palabras de aliento, las conversaciones, las risas y hasta los desahogos compartidos hicieron que este camino fuera más llevadero e inolvidable.

A los que, de una u otra forma, formaron parte de este proceso.

A todos, muchas gracias.

Índice

1. Introducción	5
2. Antecedentes	7
2.1. Propuesta	9
2.2. Hipótesis	10
2.3. Justificación y Motivación	10
2.4. Objetivos	11
2.4.1. Objetivo General	11
2.4.2. Objetivos Específicos	11
3. Estado del Arte	12
4. Marco Teórico	18
4.1. Ironía	18
4.2. Ironía en Redes Sociales	18
4.3. Aplicaciones en Análisis de Sentimientos	19
4.4. Representación del texto	20
4.4.1. <i>Bag of Words</i>	20
4.4.2. TF-IDF: <i>Term Frequency – Inverse Document Frequency</i>	21
4.4.3. BERT: <i>Bidirectional Encoder Representations from Transformers</i>	23
4.5. Técnicas de reducción de características	25
4.5.1. Análisis de Componentes Principales	25
4.6. Redes Neuronales Artificiales	27
4.7. Cómputo Evolutivo	28
4.7.1. Algoritmos Genéticos	28
4.7.2. Neuroevolución: NEAT (<i>NeuroEvolution of Augmenting Topologies</i>)	29
5. Metodología	34
5.1. Datos	34
5.2. Preprocesamiento de datos	34
5.3. Transformación de los datos con <i>Bag of Words</i> Binaria	36
5.4. Reducción con ACP	36
5.4.1. Selección de Componentes para obtener el 70 % de Varianza Explicada	37
5.5. Algoritmo Genético: Parámetros generales de NEAT	38
5.6. Procedimiento experimental con NEAT	41
6. Experimentación y Resultados	43
6.1. Experimentos	43
6.2. Entorno y herramientas a emplear	43
6.3. Resultados	44
6.3.1. Resultados de NEAT aplicando ACP	45
a. Con 50 componentes	45
b. Con 100 componentes	46
c. Con 200 componentes	47

d. 278 componentes (70 % de varianza explicada)	48
6.3.2. Resultados de NEAT con <i>Bag of Words</i> Binaria	49
a. Reducida	49
b. Original	50
6.3.3. Resultados de NEAT con TF-IDF	51
6.3.4. Comparación de desempeño de NEAT con diferentes represen- taciones de texto	52
6.3.5. Resultados de NEAT con SVM	55
7. Análisis de las Redes Evolucionadas con NEAT	57
7.1. Conexiones y Estructura de Redes Evolucionadas en NEAT	57
7.2. Distancia Genómica y Desempeño del Modelo	59
8. Conclusiones y Trabajo Futuro	63

Lista de Acrónimos

ACP	Principal Component Analysis (Análisis de Componentes Principales)
AG	Genetic Algorithm (Algoritmo Genético)
BERT	Bidirectional Encoder Representations from Transformers (Representaciones Bidireccionales de Codificadores desde Transformadores)
BoW	Bag of Words (Bolsa de Palabras)
CE	Evolutionary Computing (Cómputo Evolutivo)
CNN	Convolutional Neural Network (Red Neuronal Convolutacional)
NE	Neuroevolution (Neuroevolución)
NEAT	NeuroEvolution of Augmenting Topologies (Neuroevolución de Topologías Aumentadas)
PLN	Natural Language Processing (Procesamiento de Lenguaje Natural)
RNA	Artificial Neural Network (Red Neuronal Artificial)
SVM	Support Vector Machine (Máquinas de Soporte Vectorial)
TF-IDF	Term Frequency–Inverse Document Frequency (Frecuencia de Término – Frecuencia Inversa de Documento)

1. Introducción

El Procesamiento de Lenguaje Natural (PLN) es una rama de la Inteligencia Artificial que se ocupa del tratamiento automático del lenguaje natural, ya sea como voz o texto, brindando a las computadoras la capacidad de “comprender” textos y palabras habladas o escritas de manera análoga a los seres humanos. Para este fin, PLN combina diferentes técnicas como, por ejemplo, los modelos basados en reglas del lenguaje humano con modelos estadísticos, aprendizaje automático y aprendizaje profundo. Una de las tareas más desafiantes dentro del PLN es la *detección de ironía*, un fenómeno lingüístico que implica decir lo contrario de lo que se quiere comunicar, a menudo con una intención humorística o crítica. La detección de ironía implica no solo el reconocimiento de palabras clave o frases, sino también del conocimiento compartido entre los interlocutores, la comprensión de contextos culturales, sociales y situacionales que pueden no estar explícitamente presentes en el texto. Diversos trabajos han demostrado que el reconocimiento de este fenómeno requiere habilidades más complejas que aquellas asociadas a tareas de clasificación de texto convencionales.

La relevancia de abordar esta tarea se manifiesta en varias aplicaciones prácticas, como el análisis de sentimientos en redes sociales, donde la ironía puede distorsionar significativamente la interpretación de las opiniones [16]. En el ámbito de la interacción humano-computadora, la capacidad de detectar ironía puede llevar a comunicaciones más naturales y efectivas entre los usuarios y los sistemas basados en Inteligencia Artificial. Además, la mayoría de las técnicas convencionales requieren grandes cantidades de datos etiquetados manualmente, lo que es costoso y a menudo inviable para lenguas con pocos recursos digitales [4]. Capturar estas sutilezas requiere una comprensión profunda del lenguaje que los modelos actuales de PLN encuentran difícil de lograr. El entendimiento de la ironía requiere ir más allá del contenido textual explícito, implicando la capacidad de inferir intenciones ocultas y significados subtextuales en un contexto dado [43]. Aunque la investigación en esta área ha empleado diversas aproximaciones, desde métodos basados en reglas hasta el aprendizaje profundo, persisten desafíos en cuanto al *accuracy* —medida que indica la proporción de ejemplos clasificados correctamente y que se utilizará a lo largo de este trabajo como término de referencia principal— y la capacidad de generalización [22].

A pesar de los avances obtenidos con modelos profundos como BERT, la literatura señala que la detección automática de ironía sigue siendo una tarea abierta debido a la elevada dependencia del contexto situacional, las ambigüedades pragmáticas y la variabilidad lingüística entre usuarios de redes sociales [42]. Distintos estudios han mostrado que incluso los modelos más robustos pueden fallar cuando la ironía depende de conocimiento externo, relaciones interpersonales o referencias culturales implícitas [7]. Esta brecha entre el entendimiento humano y la interpretación computacional plantea la necesidad de explorar técnicas alternativas que permitan modelar mejor las relaciones no lineales y la variabilidad estructural del lenguaje irónico.

En este escenario, la Neuroevolución (NE) emerge como un enfoque prometedor dentro del aprendizaje automático, ya que utiliza algoritmos genéticos para optimi-

zar redes neuronales, tanto en su estructura como en los pesos de sus conexiones. A diferencia de los modelos basados en gradientes, que optimizan parámetros sobre arquitecturas fijas, los algoritmos evolutivos permiten explorar simultáneamente múltiples topologías, facilitando la búsqueda de configuraciones estructurales que se adapten de forma natural a la complejidad del fenómeno irónico [17]. Esta capacidad exploratoria resulta especialmente valiosa en dominios donde no existe un consenso claro sobre la estructura óptima del modelo, como sucede con tareas pragmáticas y altamente contextuales del PLN.

NEAT (*NeuroEvolution of Augmenting Topologies*), desarrollado por Stanley y Miikkulainen, es especialmente destacado por su capacidad para evolucionar no solo los parámetros de las redes neuronales, sino también su topología [39]. Esto permite crear modelos que se adaptan mejor a las complejidades inherentes del lenguaje irónico. NEAT ofrece ventajas únicas frente a otros esquemas neuroevolutivos, al incorporar mecanismos como la especiación, la compatibilidad genética y la expansión gradual de la topología. Estos elementos permiten preservar diversidad en la población y evitar convergencias prematuras, un problema recurrente en modelos entrenados exclusivamente con retropropagación [39]. Varios estudios en tareas lingüísticas complejas, incluyendo clasificación de discurso ofensivo y análisis de sentimientos, han señalado que los métodos evolutivos pueden descubrir arquitecturas más compactas, interpretables y robustas frente al ruido textual [26]. Esto refuerza el potencial de NEAT como una alternativa para abordar fenómenos lingüísticos que requieren comprender contradicciones semánticas, matices afectivos y disparidades pragmáticas.

El interés por integrar la Neuroevolución en la tarea de clasificación de ironía surge de la limitada aplicación de estas técnicas en el área del PLN. La Neuroevolución, como parte de las herramientas de la Computación Evolutiva (CE), se centra en optimizar tanto la topología de las redes neuronales como el ajuste de los pesos en las conexiones entre neuronas. Además, se considera que la Neuroevolución podría generar arquitecturas de redes neuronales más eficientes y adaptadas en comparación con las estructuras comúnmente utilizadas [38].

El presente trabajo se sitúa en la intersección de estas dos líneas de investigación: por un lado, el creciente interés por comprender y modelar fenómenos pragmáticos complejos; y por otro, el desarrollo de técnicas evolutivas capaces de producir arquitecturas neuronales adaptativas. Integrar diferentes representaciones vectoriales del texto —desde modelos tradicionales como Bag of Words hasta representaciones densas generadas por BERT— con un esquema evolutivo como NEAT permite evaluar empíricamente qué características estructurales favorecen la detección de ironía y cómo se comportan distintas topologías de red frente a este fenómeno. Este enfoque contribuye no solo al rendimiento en la tarea, sino también al entendimiento del vínculo entre representación lingüística, complejidad de red y desempeño predictivo.

2. Antecedentes

A lo largo de los años, se ha acumulado una cantidad significativa de investigación sobre modelos computacionales para la detección de ironía, especialmente en redes sociales como Twitter (ahora X), considerado una de las fuentes de información más utilizadas para este fin debido a su brevedad, carácter espontáneo, alta frecuencia de publicaciones, y accesibilidad a través de su API. No obstante, con el cambio de propietario y las nuevas políticas en la plataforma, las condiciones de acceso a los datos, la visibilidad de los contenidos y las dinámicas de interacción han cambiado drásticamente, lo que representa un nuevo reto para los estudios actuales y futuros en este dominio, limitando el número de peticiones diarias y eliminando el acceso público a ciertas funcionalidades, lo cual ha impactado en la recolección de datos para investigación. En el presente trabajo, se hará referencia constante al término *tweets*, en alusión a estas publicaciones breves que constituyen la unidad textual fundamental empleada en los procesos de recolección, análisis y modelado de datos.

Los primeros enfoques para la detección de ironía se enfocaron en métodos basados en reglas, donde se buscaban patrones lingüísticos específicos asociados con la ironía, como el uso de contradicciones, palabras extremas o marcadores discursivos [10]. Aunque estos métodos ofrecían una alta interpretabilidad, su aplicación era limitada debido a la falta de generalización a diferentes dominios o contextos. El estudio presentado en [36], propone un modelo multicomponente para la detección de ironía y humor en redes sociales. Su propuesta integró dimensiones como ambigüedad, polaridad y contraste semántico, evidenciando que la ironía no puede reducirse a una simple inversión del significado literal, sino que requiere la interacción de distintos niveles lingüísticos y cognitivos. Por su parte, en [37] plantearon la hipótesis de que la ironía surge a menudo del contraste entre una emoción positiva y una situación negativa. A través de un proceso de aprendizaje automático tipo *boots-trapping*, identificaron expresiones positivas (“I love”) en contextos negativos (“being ignored”), demostrando que este contraste puede servir como una señal fuerte de sarcasmo en textos cortos. De manera complementaria, el estudio de [43] enfatiza la relevancia del contexto pragmático y de la subjetividad expresada en los mensajes. Los autores argumentan que el tono irónico depende no solo del contenido literal, sino también de las intenciones comunicativas y del conocimiento compartido entre emisor y receptor, lo cual explica la dificultad que enfrentan los modelos automáticos para interpretar correctamente la ironía.

Mientras tanto, con el auge de Twitter como plataforma dominante de microblogging, surgió una nueva oleada de investigaciones centradas en el análisis de *tweets* como unidad textual básica, aprovechando su disponibilidad pública, su carácter masivo y la riqueza lingüística que ofrecen. Uno de los trabajos pioneros fue [32], donde los autores construyeron un corpus a gran escala para análisis de sentimientos a partir de *tweets*, demostrando que esta plataforma podía utilizarse como una fuente valiosa para entrenar modelos de PLN, especialmente en tareas de clasificación binaria de polaridad. Mientras tanto, en [18] se exploró un enfoque innovador al aplicar supervisión distante para etiquetar automáticamente grandes volúmenes de *tweets*, utilizando emoticonos como señales débiles de sentimiento, lo que permitió escalar la recolección de datos sin necesidad de anotadores humanos. De la misma forma, el

desarrollo de competencias internacionales impulsó considerablemente esta área de estudio. En el *SemEval-2018 Task 3*, descrito en [42], se propuso una tarea binaria centrada en la detección de ironía en *tweets* en inglés. Los sistemas participantes emplearon principalmente arquitecturas neuronales y representaciones distribucionales, alcanzando valores de F1 cercanos a 0.7, lo que evidenció avances importantes, aunque también dejó ver las limitaciones persistentes frente a la ambigüedad y la dependencia del contexto. Por otro lado, estudios más especializados han utilizado Twitter para problemas lingüísticos y sociales más complejos. Por ejemplo, en [27] se abordó la detección de discurso de odio contra comunidades específicas, mostrando cómo la brevedad y espontaneidad de los *tweets* plantea desafíos adicionales en términos de ambigüedad y contexto. Posteriormente, en [5] se exploró la dimensión semántica y pragmática de los *emojis*, analizando su papel en la interpretación del significado dentro de los *tweets*. Los autores demostraron que los *emojis* no solo funcionan como elementos decorativos o emocionales, sino que actúan como marcadores discursivos que modifican la intención comunicativa del mensaje, facilitando la transmisión de ironía, sarcasmo o humor. Su estudio evidenció que la comunicación en Twitter no se limita al texto literal, sino que incorpora múltiples capas de significado —léxicas, contextuales y paralingüísticas— que enriquecen la interacción lingüística y contribuyen a una comprensión más matizada del tono y la intención del hablante. De manera complementaria, un estudio reciente [21] abordó el desafío de la detección de ironía en Twitter bajo escenarios de distribuciones de clases desbalanceadas, una condición común en datos reales donde las instancias irónicas son considerablemente menores que las no irónicas. Su investigación propuso estrategias de reponderación y muestreo estratificado, combinadas con modelos supervisados tradicionales y representaciones distribucionales, para mitigar los efectos del desbalance en el aprendizaje. Los autores demostraron que el uso de técnicas de balanceo de clases mejora sustancialmente la capacidad de generalización de los modelos, especialmente en tareas donde los indicadores irónicos son sutiles o contextualmente dependientes. Este trabajo resulta particularmente relevante, ya que destaca la importancia de adaptar los métodos de detección de ironía a las propiedades estadísticas del corpus, una consideración esencial para investigaciones que, como la presente, buscan optimizar el desempeño del modelo en contextos con variabilidad estructural y semántica.

En paralelo, con el avance del aprendizaje automático, se comenzaron a explorar métodos basados en características extraídas de textos. Estas incluían elementos lingüísticos, sintácticos y estilísticos que alimentaban a clasificadores como Máquinas de Soporte Vectorial, Árboles de Decisión y Redes Neuronales [36]. Además, las representaciones vectoriales como *Bag of Words* (abreviada en este trabajo como BoW) facilitaron el manejo de grandes volúmenes de datos textuales, pero su capacidad para captar la complejidad de la ironía seguía siendo limitada. El surgimiento del aprendizaje profundo marcó un hito en el PLN. Modelos como las Redes Neuronales Convolucionales (CNNs) comenzaron a ser utilizados para la detección de ironía, mostrando mejoras significativas en el accuracy al capturar dependencias contextuales y patrones no lineales en los datos [44]. Posteriormente, los modelos basados en *transformers* como BERT (*Bidirectional Encoder Representations from Transformers*) ampliaron estas capacidades al proporcionar representaciones profundas contextualizadas del lenguaje [13]. Este avance supuso un punto de inflexión en el PLN, ya que permitió que los modelos aprendieran representaciones distribucionales

donde el significado de cada palabra depende del contexto circundante. Sin embargo, el aprovechamiento pleno de estas arquitecturas requiere una fase crítica de ajuste fino o *fine-tuning*, la cual adapta el modelo preentrenado a tareas específicas mediante el refinamiento de sus pesos internos. En este sentido, en el trabajo de Sun et al. [40] realizaron un análisis sobre las estrategias óptimas para el ajuste fino de BERT en tareas de clasificación de texto, demostrando que variables como la tasa de aprendizaje, el tamaño del conjunto de entrenamiento y la regularización tienen un impacto determinante en la estabilidad y desempeño final del modelo. Sus resultados establecieron un marco de referencia metodológico para el uso controlado de modelos preentrenados en dominios especializados, particularmente cuando se dispone de conjuntos de datos limitados o desbalanceados, como ocurre frecuentemente en la detección de ironía. Por su parte, en [30] profundizaron en los desafíos asociados a la estabilidad del proceso de *fine-tuning*, identificando las causas de la variabilidad observada entre ejecuciones sucesivas del mismo modelo. Los autores demostraron que factores como la inicialización aleatoria y la semilla de entrenamiento pueden inducir diferencias estadísticamente significativas en el rendimiento. Estas observaciones son particularmente relevantes para enfoques que, como el presente trabajo, involucran procesos evolutivos o estocásticos, ya que enfatizan la necesidad de evaluar la robustez de los resultados bajo múltiples ejecuciones y condiciones iniciales controladas.

En tanto, la computación evolutiva ha comenzado a ser explorada como una alternativa innovadora para abordar las limitaciones de los enfoques tradicionales. Investigaciones como las de Stanley et al. [38] sugieren que la aplicación de estos métodos en tareas complejas de clasificación, incluida la detección de ironía, puede superar las limitaciones de las aproximaciones convencionales. En los capítulos siguientes se describen la propuesta metodológica, la hipótesis y los objetivos que sustentan y justifican el desarrollo de este trabajo, así como el diseño experimental que permitió evaluar el potencial de los algoritmos evolutivos aplicados a la detección automática de ironía.

2.1. Propuesta

La presente investigación propone el uso de NEAT como una alternativa innovadora y eficaz para la detección de ironía en texto, abordando las limitaciones de los modelos tradicionales de aprendizaje supervisado mediante la evolución automática de arquitecturas de redes neuronales.

El enfoque planteado consiste en evolucionar redes neuronales mediante NEAT utilizando diferentes representaciones del texto, tales como *Bag of Words* (con peso binario y TF-IDF) y BERT, y aplicar técnicas de reducción de dimensionalidad, como el Análisis de Componentes Principales (ACP), para mejorar la eficiencia y estabilidad del modelo. A diferencia de los modelos tradicionales basados en aprendizaje supervisado, que emplean arquitecturas fijas y dependen en gran medida de un ajuste manual de hiperparámetros, NEAT permite una exploración automática del espacio de soluciones, lo que contribuye a obtener redes neuronales potencialmente mejor adaptadas al problema de clasificación. Además de la evolución topológica de redes neuronales, se propone un análisis estructural detallado que abarca tanto las conexiones habilitadas en las redes finales como la distancia genómica entre el genoma ganador y los demás individuos de la población. Este análisis permitirá ob-

tener una mayor interpretabilidad del modelo, aportando una visión profunda sobre el papel que juegan las distintas características en la detección de ironía.

2.2. Hipótesis

El uso de NEAT para la evolución de redes neuronales, permitirá mejorar la detección de ironía, logrando al menos un desempeño similar al de los modelos tradicionales al abordar de manera más efectiva las sutilezas de esta figura del lenguaje.

2.3. Justificación y Motivación

La motivación central de este trabajo radica en explorar los detalles relacionados con las limitaciones actuales en la clasificación de ironía mediante la combinación de representaciones textuales avanzadas y enfoques evolutivos. La investigación existente en la detección de ironía ha utilizado una variedad de técnicas, y a menudo estos métodos se enfrentan a limitaciones, especialmente cuando se trata de textos cortos como *tweets*, donde el contexto limitado aumenta la dificultad de la tarea [16].

La elección de representaciones textuales desempeña un papel crítico en la capacidad de los modelos de PLN para procesar y comprender la ironía. Representaciones clásicas como *Bag of Words* han sido ampliamente utilizadas debido a su simplicidad y eficacia en tareas básicas. Sin embargo, estas aproximaciones tienden a ser insuficientes para capturar relaciones contextuales profundas y significados implícitos. Por otro lado, enfoques más avanzados, como el ACP, permiten reducir la dimensionalidad de los datos, mejorando la interpretabilidad sin perder información relevante.

Recientemente, las representaciones basadas en modelos de *transformers* como BERT han demostrado ser particularmente útiles para capturar patrones contextuales y semánticos complejos. BERT proporciona una representación contextualizada de un texto, lo que lo hace especialmente adecuado para tareas que requieren un entendimiento profundo, como la detección de ironía. Esta capacidad de integrar el contexto de manera bidireccional resulta crucial para desentrañar las intenciones irónicas en textos.

NEAT ofrece una metodología prometedora para optimizar la estructura de las redes neuronales, permitiendo la evolución de modelos que pueden adaptarse mejor a las particularidades del lenguaje irónico. Una mejora clave que se propone es integrar representaciones como BERT y ACP en la fase de entrenamiento de NEAT. Al hacerlo, se busca no solo optimizar los pesos de las conexiones neuronales, sino también proporcionar al algoritmo datos enriquecidos y representaciones robustas que faciliten la clasificación de la ironía. Además, se plantea el diseño de estrategias para mejorar la interpretabilidad de las redes generadas por NEAT. Esto incluye analizar las conexiones y nodos relevantes, así como evaluar la contribución de diferentes representaciones textuales en el desempeño del modelo. Este enfoque no solo mejora la eficacia del modelo, sino que también proporciona una mayor comprensión de los factores que influyen en la detección de ironía.

Finalmente, al evaluar el algoritmo de NEAT en este contexto, se espera contribuir al desarrollo de soluciones más interpretables y generalizables para tareas

complejas de PLN. Este enfoque tiene el potencial de transformar cómo se abordan los matices lingüísticos en el PLN, proporcionando herramientas más potentes y flexibles para comprender el lenguaje humano.

2.4. Objetivos

2.4.1. Objetivo General

Usar NEAT para evolucionar redes neuronales para la detección de ironía en textos, explorando distintas representaciones del lenguaje e incluyendo transformaciones mediante ACP con el propósito de explorar su posible capacidad de mejorar el desempeño de esta tarea con relación a otras técnicas.

2.4.2. Objetivos Específicos

- Diseñar y entrenar redes neuronales mediante NEAT que optimicen tanto la topología como los pesos de las conexiones neuronales para la detección de ironía en textos.
- Comparar el desempeño del modelo basado en NEAT con métodos tradicionales de aprendizaje supervisado, considerando métricas de accuracy, así como su adaptabilidad y capacidad de generalización.
- Evaluar el impacto de las representaciones textuales, como *Bag of Words*, y BERT, en el desempeño del modelo propuesto.
- Proporcionar interpretabilidad al modelo mediante el análisis de las conexiones neuronales y características clave identificadas durante el proceso de evolución.

3. Estado del Arte

La ironía ha sido estudiada desde diversas perspectivas en lingüística, filosofía del lenguaje y ciencias cognitivas, considerándose una figura retórica que implica una disociación entre el significado literal y la intención comunicativa. Para [2], se trata de un fenómeno complejo que involucra inferencias contextuales, conocimiento compartido y marcadores lingüísticos sutiles. En este sentido, puede concebirse como una forma de “pretensión” comunicativa, en la cual el hablante transmite una intención opuesta a la literalidad del mensaje [11]. Diversos trabajos han propuesto enfoques supervisados para abordar esta tarea, evidenciando que la inclusión de variables afectivas mejora significativamente el rendimiento de los modelos. En este sentido, en [16] se propuso un enfoque basado en el uso de características lingüísticas y afectivas derivadas de recursos léxicos como SenticNet y WordNet-Affect [3], complementadas con representaciones distribucionales obtenidas a partir de word embeddings. Dicho estudio demostró que la ironía, al involucrar un desbalance entre la carga emocional y el contenido semántico literal del texto, puede capturarse con mayor precisión mediante la integración de descriptores afectivos que reflejan la polaridad y la intensidad emocional del discurso. Por su parte, en [15] se amplió esta línea de investigación al incorporar features semánticas y afectivas dentro de un marco supervisado de clasificación binaria, empleando algoritmos como SVM y RNA. Su contribución radicó en demostrar que la ironía no sólo se relaciona con la presencia de palabras afectivamente contrastantes, sino también con la configuración contextual en la que aparecen. En particular, el estudio resaltó el papel de la incongruencia entre las emociones implícitas y las explícitas expresadas en el texto, estableciendo que los modelos que integran representaciones emocionales, semánticas y de polaridad logran capturar con mayor efectividad el carácter sutil y contextual de la ironía. Además de los modelos supervisados, se exploraron enfoques basados en reglas y patrones explícitos, como en [41], donde se propuso un sistema que identificaba expresiones irónicas a partir de la presencia de emoticonos, comillas o marcadores discursivos, tales como comillas, exclamaciones repetidas o interjecciones exageradas. Por su parte, en [12] se presentó un enfoque más flexible basado en la extracción automática de patrones discriminativos desde grandes corpus de *tweets*, combinando información léxica, sintáctica y de puntuación con la presencia de emoticonos y hashtags. Este método permitió descubrir plantillas irónicas sin necesidad de reglas predefinidas, aprovechando la frecuencia y coocurrencia de términos en contextos irónicos y no irónicos.

La mayoría de los estudios sobre ironía han sido realizados en Inglés. Sin embargo, han surgido iniciativas para crear corpus en Español, como lo muestra el trabajo presentando en [31], donde los autores lideraron la tarea de detección de ironía en variantes del Español en IberLEF 2019. Estos esfuerzos permiten evaluar la aplicabilidad de distintos métodos en contextos lingüísticos específicos. En este mismo sentido, un referente fundamental en la investigación sobre detección de ironía es el *SemEval-2018 Task 3: Irony Detection in English Tweets*, organizado por Van Hee, Lefever y Hoste [42]. En esta competencia se abordó la tarea binaria de determinar si un tweet contenía ironía o no (Task A), utilizando un corpus balanceado de 4,792 instancias, dividido en conjuntos de entrenamiento y prueba. Los detalles de este corpus se presentan en la sección de Metodología (véase Sección 5), específicamente

en la subsección de Datos. Para la evaluación de los sistemas se emplearon métricas estándar en PLN: *accuracy* (proporción global de aciertos), *precision* (proporción de verdaderos positivos entre los predichos como positivos), *recall* (proporción de verdaderos positivos detectados respecto al total real de positivos) y *F1-score* (media armónica entre *precision* y *recall*). En total participaron 43 equipos, de los cuales el mejor desempeño fue alcanzado por THU-NGN, con un F1 de 0.705, lo que evidenció un adecuado equilibrio entre *precision* (0.630) y *recall* (0.801). Resulta interesante que los sistemas restringidos al corpus oficial lograran incluso mejores resultados que aquellos que incorporaron datos externos, como lo demostró el equipo UCDDC, que obtuvo un F1 de 0.724 y la mayor precisión del conjunto (0.788). En contraste, los sistemas no restringidos, que podían utilizar recursos adicionales, no superaron a los mejores restringidos, siendo el de #NonDicevo-SulSerio el mejor con un F1 de 0.622. Este hallazgo sugiere que la calidad de la anotación y el modelado del corpus base puede ser más determinante que la cantidad de datos disponibles. En conjunto, los resultados de la Task A mostraron que, si bien la detección binaria de ironía es abordable mediante arquitecturas profundas y embeddings preentrenados, con desempeños cercanos a un F1 de 0.70, aún persiste un margen de error significativo.

Uno de los pilares fundamentales para la detección automática de ironía es la representación vectorial del texto. Entre las más tradicionales se encuentran BoW, TF-IDF, y los vectores distribucionales obtenidos mediante Word2Vec [29]. Si bien estas técnicas han sido útiles para tareas generales de clasificación, presentan limitaciones importantes para capturar relaciones profundas, necesarias en el análisis de ironía. Más recientemente, los modelos basados en *transformers* como BERT han mostrado un desempeño sobresaliente en tareas de clasificación, al capturar relaciones contextuales profundas en el lenguaje [13]. Por ejemplo, en [34] se aplicaron modelos basados en BERT y RoBERTa para la detección de sarcasmo e ironía en Twitter. En su estudio, demostraron que estas arquitecturas, al aprovechar mecanismos de atención bidireccional y representaciones contextuales profundas, superan consistentemente a las técnicas tradicionales de representación vectorial como BoW y TF-IDF. Además, su investigación puso de manifiesto que los modelos *transformer* son capaces de capturar dependencias semánticas de largo alcance y matices pragmáticos en el discurso. Este enfoque consolidó la transición hacia técnicas basadas en aprendizaje profundo, marcando un precedente en el uso de representaciones lingüísticas dinámicas que adaptan el significado de cada palabra en función de su contexto.

Mientras tanto, en el campo del análisis de sentimientos y la detección de ironía, estudios como el de [19], realizaron uno de los primeros análisis detallados sobre la detección de sarcasmo en Twitter. Compararon diferentes modelos de clasificación, como SVM y Naive Bayes, bajo escenarios con distintos niveles de balanceo entre clases, evidenciando que el desbalance de clases afecta significativamente la capacidad del modelo para reconocer adecuadamente el sarcasmo. Entre sus hallazgos destaca que los modelos tienden a sobreajustarse a la clase mayoritaria, por lo que es necesario diseñar estrategias específicas de muestreo o de ponderación de clases. Este punto resulta especialmente relevante para el presente trabajo, dado que la ironía y el sarcasmo suelen aparecer con menor frecuencia que el lenguaje literal, lo que conduce a distribuciones desbalanceadas que deben ser tratadas cuidadosamente para evitar sesgos en el entrenamiento. En conjunto, el estudio proporciona

una base empírica sólida que justifica la incorporación de *features* pragmáticas y el tratamiento explícito del desbalance, aspectos que se retoman en investigaciones posteriores tanto con modelos supervisados tradicionales como con enfoques basados en representaciones profundas. Otros trabajos han extendido estas ideas incorporando representaciones distribucionales y modelos neuronales más complejos. Por ejemplo, Bamman y Smith [4] exploraron el sarcasmo en Twitter desde una perspectiva socio-lingüística, modelando no solo el contenido textual, sino también información del usuario y del destinatario, con el fin de capturar patrones de uso idiosincrásicos. Sus resultados mostraron que el sarcasmo está fuertemente ligado a la identidad y al historial de interacción de los usuarios, lo que sugiere que los modelos de detección pueden beneficiarse de incorporar información de perfil y contexto social, más allá del texto aislado. De manera complementaria, estudios como los de Barbieri et al. [5] analizaron la relación entre emojis, marcadores estilísticos y fenómenos pragmáticos como la ironía, evidenciando que la señal paralingüística (por ejemplo, emojis ambiguos o combinaciones inusuales de iconos) aporta pistas adicionales sobre la intención del hablante que no se capturan únicamente con n-gramas de palabras. Posteriormente, en [36] se propuso un enfoque multidimensional para la detección de ironía que integraba dimensiones semánticas, emocionales y estilísticas. Los autores demostraron que la ironía no puede capturarse de manera efectiva con un solo tipo de representación, sino que requiere la combinación de múltiples niveles de información lingüística. Este hallazgo anticipó la relevancia de las representaciones contextuales modernas, como las generadas por modelos basados en *transformers* (por ejemplo, BERT), las cuales son capaces de codificar relaciones de dependencia entre palabras en función del contexto en el que aparecen.

En lo que respecta al modelado, la computación evolutiva ha sido empleada en problemas de clasificación mediante algoritmos como los genéticos, estrategias evolutivas y algoritmos híbridos. En [14], se muestra una de las referencias más completas sobre los fundamentos teóricos y prácticos de la computación evolutiva, ofreciendo un marco unificado que abarca desde los principios biológicos que inspiran los algoritmos evolutivos hasta su formalización matemática y sus aplicaciones prácticas. Los autores presentan una clasificación exhaustiva de los principales paradigmas evolutivos, incluyendo los algoritmos genéticos, las estrategias evolutivas, la programación genética y los sistemas de evolución diferencial, discutiendo sus fortalezas, limitaciones y áreas de aplicación dentro del aprendizaje automático, la optimización multimodal, la ingeniería y la inteligencia artificial. Un aspecto particularmente relevante del libro es su énfasis en cómo la CE permite abordar problemas donde los métodos tradicionales resultan ineficientes debido a la no convexidad, la alta dimensionalidad o la ausencia de gradientes útiles. Este nivel de sistematización convierte a la obra en una referencia clave para comprender cómo los métodos evolutivos pueden extenderse a escenarios más complejos, como la optimización de arquitecturas de redes neuronales, la búsqueda en espacios estructurales y la adaptación dinámica de modelos, aspectos que sirven de fundamento conceptual para enfoques modernos como NEAT y otras variantes de neuroevolución.

Previamente, los trabajos pioneros de Holland en [23] sentaron las bases conceptuales de los algoritmos genéticos, introduciendo la idea de que los procesos de evolución biológica pueden ser simulados computacionalmente para explorar espacios

de búsqueda complejos y adaptativos. En esta obra, Holland no solo propone la idea de simular mecanismos de selección natural para resolver problemas computacionales, sino que desarrolla un marco teórico sólido que explica por qué y cómo estos algoritmos funcionan. Su formulación de los operadores genéticos (cruza, mutación y selección), junto con la representación basada en cadenas binarias, estableció los principios metodológicos que posteriormente inspirarían múltiples variantes evolutivas. Uno de sus aportes más influyentes es la *Teoría de Esquemas*, la cual demuestra que los algoritmos genéticos pueden considerarse como un mecanismo de procesamiento paralelo implícito, capaz de explorar simultáneamente múltiples regiones del espacio de búsqueda. Este marco teórico introduce la noción de bloques constructores: combinaciones de características parciales que, al ser preservadas y recombinadas por el proceso evolutivo, permiten construir soluciones cada vez más complejas y adaptadas. Asimismo, Holland destaca la relevancia del equilibrio entre exploración y explotación, principio que ha guiado durante décadas el diseño de algoritmos evolutivos y que continúa siendo esencial en enfoques modernos como la neuroevolución. Al conectar los mecanismos evolutivos con modelos de adaptación en sistemas biológicos, cognitivos y computacionales, su trabajo posiciona a los AG como herramientas capaces de enfrentar problemas donde los métodos deterministas tradicionales fallan, debido a su capacidad para navegar espacios altamente no lineales, multimodales o ruidosos.

Más recientemente, en [26] se presentó un análisis actualizado sobre el uso de neuroevolución y algoritmos genéticos aplicados a la clasificación de texto, destacando cómo la evolución de arquitecturas neuronales permite optimizar tanto los pesos como la topología de las redes. Los autores demostraron que estos métodos pueden obtener resultados competitivos frente a enfoques de aprendizaje profundo, especialmente en escenarios donde los recursos de entrenamiento o el tamaño del conjunto de datos son limitados. Estos enfoques permiten explorar espacios de soluciones no lineales, adaptativos y robustos ante ruido o ambigüedad. En PLN, su aplicación ha cobrado relevancia, particularmente en tareas como clasificación de sentimientos, detección de discurso de odio [44], e incluso ironía [22]. Por ejemplo, en [9] se desarrolló un modelo basado en atención jerárquica para identificar ironía en publicaciones de redes sociales, destacando que la incorporación explícita de contexto discursivo mejora el rendimiento sobre métodos secuenciales tradicionales. Este enfoque surge como respuesta a la limitación de los modelos secuenciales tradicionales —como LSTM o GRU—, los cuales suelen perder información relevante cuando el fenómeno irónico depende de relaciones de largo alcance entre partes distantes del texto. El modelo presentado por los autores incorpora un mecanismo de fusión jerárquica multimodal, integrando no solo la señal lingüística, sino también elementos visuales provenientes de imágenes adjuntas a las publicaciones en Twitter. Esta integración resulta especialmente valiosa debido a que, en contextos reales de redes sociales, la ironía suele manifestarse a través de la combinación de texto e imágenes, las cuales pueden amplificar o contradecir el contenido verbal de manera intencional. Los resultados experimentales muestran que este modelo supera significativamente a enfoques puramente textuales, lo que refuerza la importancia del contexto discursivo ampliado en la interpretación de fenómenos pragmáticos como la ironía. El trabajo demuestra que los matices irónicos pueden depender de señales no literales que emergen de la interacción entre texto, tono implícito y elementos visuales, por lo que su modelado requiere mecanismos capaces de integrar múltiples niveles de representación.

En ese sentido, uno de los algoritmos más influyentes en este campo es NEAT. En este trabajo, el aspecto innovador es el análisis estructural de las redes neuronales generadas por dicho algoritmo. En particular, se analiza la activación de conexiones habilitadas y deshabilitadas hacia la capa de salida, así como la distancia genómica entre los genomas de la población, para entender mejor la dinámica evolutiva del modelo. Estos análisis permiten evaluar no solo la **convergencia** del algoritmo —entendida como el proceso mediante el cual la población evolutiva alcanza estabilidad en su desempeño, acercándose a una solución óptima—, sino también la relevancia de ciertas características del input textual y su impacto en la arquitectura final. También, se exploraron múltiples representaciones de texto: BoW binaria, BoW reducida por frecuencia, TF-IDF, embeddings de BERT y reducción de dimensionalidad mediante ACP. Dado que el uso de modelos complejos incrementa los requerimientos computacionales, se han incorporado técnicas de reducción de dimensionalidad, siendo ACP una de las más utilizadas. En [1, 25], se ofrece una revisión exhaustiva sobre los fundamentos teóricos y las aplicaciones modernas de esta técnica, así como de sus avances recientes, destacando su capacidad para identificar estructuras latentes, eliminar redundancias y capturar la mayor parte de la varianza presente en los datos mediante un conjunto reducido de componentes ortogonales. Su revisión enfatiza cómo el ACP facilita la interpretación de las relaciones subyacentes entre variables, además de servir como paso previo para técnicas supervisadas y no supervisadas. Por su parte, en [6] se subraya que la reducción de dimensionalidad mediante ACP no solo contribuye a disminuir el ruido y la redundancia en los datos, sino que también favorece la estabilización del entrenamiento y mejora la capacidad de generalización de los modelos de aprendizaje, tanto en clasificadores tradicionales como en enfoques evolutivos. En este contexto, la aplicación del ACP en el presente trabajo permitió equilibrar la complejidad computacional de los modelos evolutivos con la retención de información relevante para la detección de ironía.

A partir del análisis de los estudios revisados, es posible observar que la detección automática de ironía continúa siendo un campo en desarrollo dentro del PLN. La mayoría de las investigaciones recientes se han centrado en la utilización de modelos basados en *transformers* o en métodos supervisados tradicionales, con resultados prometedores en lenguas como el inglés. Sin embargo, se tiene una menor atención hacia la aplicación de enfoques evolutivos, los cuales podrían ofrecer ventajas en cuanto a la exploración estructural y la adaptabilidad del modelo ante la ambigüedad inherente a la ironía. La Tabla 1 resume los principales trabajos identificados que abordan la detección de ironía, destacando sus metodologías, contribuciones y limitaciones. Esta comparación permite contextualizar las aportaciones del presente estudio dentro del panorama actual.

Tabla 1: Comparación de trabajos relacionados con la detección de ironía y enfoques afines.

Autores	Metodología y principales contribuciones	Limitaciones
Tepperman <i>et al.</i> (2006) [41]	Sistema de reglas basado en marcadores explícitos (puntuación, comillas, énfasis, interjecciones). Identifica señales superficiales que correlacionan con ironía.	Baja capacidad para detectar ironía implícita o dependiente del contexto discursivo.
Davidov <i>et al.</i> (2010) [12]	Extracción automática de patrones sarcásticos mediante coocurrencia de hashtags, emoticonos y términos. Permite la identificación semisupervisada de sarcasmo.	Dependencia de señales explícitas y del dominio; limitada generalización.
González-Ibáñez <i>et al.</i> (2011) [19]	Análisis de sarcasmo en Twitter mediante <i>features</i> léxicas, pragmáticas y contextuales. Destaca el papel del contexto y las señales pragmáticas en la interpretación irónica.	Modelos poco profundos; incapaces de capturar dependencias complejas.
Bamman y Smith (2015) [4]	Detección contextualizada de sarcasmo en Twitter mediante la incorporación explícita de información del autor, su historial lingüístico y las relaciones sociales dentro de la plataforma. El modelo integra características pragmáticas y metadatos para mejorar la comprensión del contexto, mostrando que el sarcasmo requiere información extratextual para ser detectado con mayor precisión.	Limitado a sarcasmo (no ironía en general); el enfoque depende fuertemente de metadatos y estructura social, lo que dificulta su aplicación en corpus sin información contextual adicional.
Hernández-Farías <i>et al.</i> (2016) [16]	Clasificación supervisada basada en <i>features</i> lingüísticas, afectivas y representaciones distribucionales (<i>embeddings</i>). Demuestra que la combinación de información semántica y emocional mejora la detección de ironía frente a modelos puramente léxicos.	Alta dependencia de recursos externos; generalización limitada a otros dominios e idiomas.
Reyes, Rosso y Veale (2016) [36]	Modelo multidimensional que integra información semántica, emocional y estilística. Demuestra que la combinación de distintas dimensiones lingüísticas mejora la detección de ironía.	Proceso de extracción complejo; limitada escalabilidad a grandes corpus.
Hernández (2017) [15]	Uso de SVM y RNA con variables semánticas y afectivas. Evidencia la importancia de la incongruencia entre emoción y literalidad del texto como indicador de ironía.	Sensibilidad a la calidad del etiquetado; requiere ingeniería manual de características.
Van Hee, Lefever y Hoste (2018) [42]	Competencia SemEval 2018 para ironía en inglés. Evalúa sistemas basados en SVM, redes neuronales y primeros modelos <i>transformer</i> . Establece un benchmark estandarizado para la tarea.	Los mejores modelos alcanzan $F1 \approx 0.70$, reflejando la dificultad persistente del fenómeno.
Bueno <i>et al.</i> (2019) [31]	Desarrollo del corpus IberLEF 2019 para detección de ironía en variantes del español. Presenta IroSvA, la primera tarea compartida para identificar la presencia de ironía en mensajes cortos. Propone evaluación comparativa entre modelos clásicos y redes neuronales.	Tamaño reducido del corpus; predominio de enfoques tradicionales.
Stanley (2019) [38]	Propuesta del algoritmo NEAT para evolución conjunta de topología y pesos. Sienta las bases de la neuroevolución moderna y su aplicación en problemas de aprendizaje.	Aplicación limitada en PLN; no enfocado directamente en ironía.
Potamias, Siolas y Stafylopatis (2020) [34]	Aplicación de BERT y RoBERTa para detección de ironía y sarcasmo en Twitter. Los modelos <i>transformer</i> superan significativamente a BoW y TF-IDF.	Requiere grandes volúmenes de datos y alta capacidad computacional.
Hernández <i>et al.</i> (2021) [22]	Aplicación de cómputo evolutivo para el análisis de sentimientos en <i>tweets</i> escritos en español mexicano. Evalúa variantes evolutivas como alternativa a redes profundas tradicionales.	Resultados preliminares; falta de análisis en topologías neuronales.
Kadamala y Griffith (2023) [26]	Uso de neuroevolución y algoritmos genéticos para clasificación de texto. Explora la optimización evolutiva de arquitecturas neuronales y pesos.	No enfocado directamente en ironía; sin análisis estructural profundo.

4. Marco Teórico

En esta sección se presentan los conceptos fundamentales que sustentan este trabajo. Se aborda la definición de ironía y su manifestación en redes sociales, especialmente en Twitter, así como su relación con el análisis de sentimientos. También se revisan técnicas de representación vectorial de texto y los fundamentos de las redes neuronales artificiales y del algoritmo NEAT, que constituye la base metodológica de esta investigación.

4.1. Ironía

De acuerdo con la Real Academia Española (RAE), la ironía es una figura retórica que expresa algo contrario o diferente a lo que literalmente se dice, generalmente con un propósito de burla disimulada. En particular, la ironía está estrechamente vinculada a fenómenos afectivos y a la expresión de sentimientos implícitos, lo que refuerza su interpretación subjetiva [15]. Esta figura del lenguaje se caracteriza por el uso de recursos lingüísticos como contradicciones, exageraciones, omisiones y expresiones inesperadas que generan un contraste cognitivo [36]. Además, la ironía suele dividirse en dos categorías principales: verbal y situacional. La ironía verbal implica una contradicción directa entre el significado literal y el intencionado, mientras que la situacional ocurre cuando existe una incongruencia entre las expectativas y la realidad. En redes sociales, la mayoría de los estudios se han enfocado en la ironía verbal debido a su frecuencia en los textos generados por los usuarios [36].

4.2. Ironía en Redes Sociales

Las redes sociales, son entornos ideales para el estudio de la ironía debido a su estilo comunicativo breve y creativo. Los usuarios de estas plataformas a menudo emplean hashtags como *#irony* o *#sarcasm* para etiquetar manualmente expresiones irónicas o respectivamente sarcásticas, lo que facilita la recopilación de datos para el entrenamiento de modelos de PLN [15]. Sin embargo, esta autoetiquetación también introduce ruido, ya que no todos los textos etiquetados reflejan estrictamente ironía o sarcasmo según definiciones formales.

Estudios han identificado patrones estilísticos comúnmente asociados con la ironía en redes sociales, como el uso de puntuación especial, emoticonos y estructuras sintácticas complejas [10]. Además, los *tweets* irónicos suelen contener palabras con polaridades extremas que refuerzan el contraste implícito del mensaje.

Un ejemplo claro de la variedad de usos irónicos en redes sociales se puede observar en el siguiente conjunto de publicaciones tomadas de Twitter, Reddit y reseñas de productos en Amazon, citadas en [15]:

1. *“Dad is giving me pointers on how to grill when he burns almost everything he grills #irony”.*
2. *“It’s totally fine to blare music in the house when I have school in the morning... #sarcasm”.*
3. *“I honestly do not know if this article is for real... no one is that stupid... right??”.*

4. *"I cannot recommend this monitor to anyone, unless you enjoy not sleeping - not due to a baby crying, but rather your monitor beeping".*

Los primeros intentos para detectar ironía utilizaron métodos basados en reglas lingüísticas y patrones predefinidos. Por ejemplo, se emplearon expresiones como onomatopeyas, marcas de puntuación y uso de comillas para identificar posibles mensajes irónicos [10]. Aunque estos enfoques eran interpretables, presentaban limitaciones significativas en su capacidad para generalizar a contextos más diversos. Con la adopción del aprendizaje automático, los enfoques para la clasificación de ironía comenzaron a incorporar representaciones vectoriales. Estudios como los de [36] propusieron modelos multidimensionales que integraban elementos como gramáticas de estilo, escenarios emocionales y estructuras textuales para mejorar la detección de ironía.

La incorporación de información afectiva ha demostrado ser particularmente útil para la detección de ironía. El modelo emotIDM, desarrollado en [15], utiliza características relacionadas con las emociones y la polaridad para mejorar la clasificación de *tweets* irónicos. Este enfoque ha mostrado que las emociones juegan un papel crucial en la distinción entre mensajes literales e irónicos, y ofrece un marco prometedor para futuras investigaciones.

La interpretación de la ironía en redes sociales y plataformas digitales depende en gran medida del conocimiento compartido entre los interlocutores y del uso de recursos lingüísticos y pragmáticos que permiten inferir significados implícitos [8]. Estos elementos facilitan la identificación de mensajes irónicos, pero también plantean desafíos para los sistemas automáticos de procesamiento del lenguaje, que deben diferenciar entre el contenido irónico y el literal en ausencia de contexto. También destacan la importancia de considerar la ironía como una estrategia comunicativa que permite a los usuarios expresar críticas, desacuerdos o emociones de manera indirecta, lo que contribuye a su prevalencia en debates políticos, análisis de productos y comentarios sociales. Este enfoque refuerza la necesidad de desarrollar modelos de PLN que no solo detecten la ironía, sino que también comprendan su propósito comunicativo y su impacto en la interacción social.

4.3. Aplicaciones en Análisis de Sentimientos

La detección de ironía tiene aplicaciones significativas en el campo del análisis de sentimientos, donde su identificación puede mejorar el accuracy y relevancia de los modelos en contextos clave. Algunas de las aplicaciones principales incluyen:

- **Servicios de monitoreo de redes sociales:** Estos servicios pueden beneficiarse de la detección de ironía al filtrar y clasificar contenido de interés. Por ejemplo, comentarios que poseen un significado diferente al literal podrían ser correctamente identificados. Esto es especialmente relevante para propósitos de marketing y políticos, donde malinterpretar mensajes irónicos podría impactar negativamente en la percepción de éxito o progreso de una campaña.
- **Predicción de movimientos del mercado bursátil:** Se ha demostrado que los sentimientos expresados en textos de redes sociales influyen en la predicción de movimientos del mercado bursátil. Para analizar cómo las opiniones de las

personas afectan al mercado, es crucial contar con indicadores confiables del sentimiento en el contenido social. La presencia de ironía podría socavar la interpretación precisa de estos resultados.

- **Predicción de tendencias sociales:** El contenido irónico puede proporcionar información valiosa sobre actitudes sociales y culturales en temas específicos, como política o eventos actuales. Incorporar la detección de ironía en modelos de análisis de tendencias puede ofrecer una visión más precisa y detallada de la opinión pública [36].
- **Sistemas de recomendación:** En plataformas como redes sociales y sitios de reseñas, identificar la ironía ayuda a mejorar la calidad de las recomendaciones, garantizando que el contenido irónico no sesgue los algoritmos de recomendación hacia resultados incorrectos [15].

Diversos enfoques han sido propuestos para abordar los desafíos asociados con la clasificación de ironía. Las siguientes secciones exploran las diferentes representaciones textuales, así como las técnicas propuestas e implementadas en este trabajo. Estas metodologías buscan optimizar el análisis de ironía mediante la integración de representaciones contextuales avanzadas y enfoques computacionales innovadores.

4.4. Representación del texto

La representación del texto es un componente clave en el campo del PLN, pues permite convertir datos textuales en estructuras numéricas procesables por algoritmos computacionales. Esto se debe a que las computadoras no entienden directamente palabras o símbolos lingüísticos; en cambio, operan eficientemente con valores numéricos.

4.4.1. *Bag of Words*

El modelo de *Bag of Words* (BoW) se basa en la idea de ignorar el orden de las palabras en un texto, representándolo únicamente como un conjunto (o “bolsa”) de palabras con su frecuencia asociada. Este modelo, introducido como una de las primeras estrategias en PLN, ofrece una representación simplificada que, aunque limitada en algunos aspectos, sigue siendo relevante en problemas de clasificación y minería de texto.

El modelo de BoW opera bajo un principio básico: cada documento se representa como un vector de palabras, donde el orden de los términos es completamente ignorado. En su lugar, se enfoca en la frecuencia con la que cada palabra aparece dentro de un documento específico. El proceso para construir esta representación incluye pasos como la tokenización del texto, la creación de un vocabulario único basado en el corpus y la vectorización que asigna valores numéricos a las palabras en función de su ocurrencia.

Dada una frase, como “La luna brilla y la noche es hermosa”, el modelo generará un vocabulario con términos como [“La”, “luna”, “brilla”, “y”, “noche”, “es”, “hermosa”] y representará cada frase como un vector binario o ponderado según las ocurrencias de esas palabras.

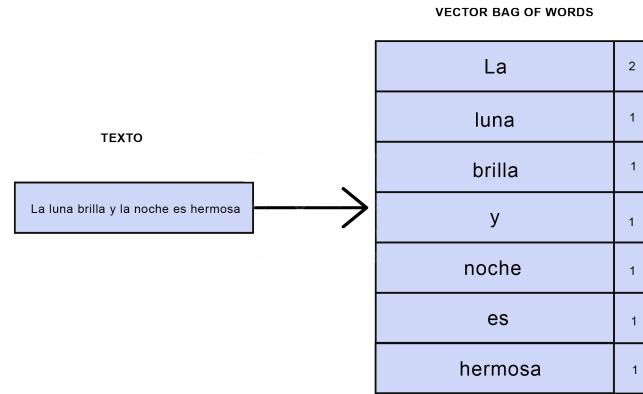


Figura 1: Representación Vectorial de *Bag of Words*: Ejemplo Práctico. Fuente: Elaboración propia.

Una de las principales ventajas del modelo BoW es su simplicidad, lo que facilita su implementación en una variedad de contextos. Además, es flexible y eficiente desde el punto de vista computacional, lo que lo hace adecuado para tareas como clasificación de texto, análisis de sentimientos y detección de spam [33]. Sin embargo, su simplicidad también trae consigo limitaciones importantes, como la incapacidad de capturar el orden o el contexto de las palabras, así como el problema de alta dimensionalidad cuando el vocabulario del corpus es extenso.

BoW es a menudo utilizado como un punto de partida en el análisis de texto, especialmente en tareas exploratorias o en contextos donde la simplicidad y la interpretabilidad son más importantes que el rendimiento predictivo avanzado. Aunque técnicas más recientes, como Word2Vec o BERT, han ganado popularidad por su capacidad para capturar relaciones semánticas profundas, BoW sigue siendo una herramienta fundamental en el PLN, proporcionando una base sólida para el desarrollo de modelos más complejos [13, 29].

4.4.2. TF-IDF: *Term Frequency – Inverse Document Frequency*

El modelo *TF-IDF* (*Term Frequency - Inverse Document Frequency*) es una técnica fundamental para la representación de texto y la recuperación de información. A diferencia del modelo BoW, que simplemente contabiliza la frecuencia de las palabras en un documento, *TF-IDF* introduce un enfoque ponderado que evalúa la importancia de un término no solo dentro de un documento, sino también en relación con el resto del corpus [35].

El modelo *TF-IDF* combina dos métricas principales: la frecuencia de término (*TF*) y la frecuencia inversa de documento (*IDF*). Estas métricas se combinan multiplicativamente para calcular el peso de cada término en un documento.

La frecuencia de término mide cuántas veces aparece una palabra en un docu-

mento en relación con el total de palabras de ese documento. TF se define como en la ecuación (1), donde:

$$TF(t, d) = \frac{f_{t,d}}{\sum_k f_{k,d}} \quad (1)$$

Donde:

- $f_{t,d}$ es el número de ocurrencias del término t en el documento d .
- $\sum_k f_{k,d}$ es el total de palabras en el documento d .

Por su parte, IDF mide la rareza de un término en el corpus. Se calcula con la ecuación (2):

$$IDF(t, D) = \log \left(\frac{N}{1 + |d \in D : t \in d|} \right) \quad (2)$$

Donde:

- N es el número total de documentos en el corpus D .
- $|d \in D : t \in d|$ es el número de documentos en los que aparece el término t .

Se suma 1 en el denominador para evitar divisiones por cero.

La combinación de ambas métricas se realiza como en la ecuación (3):

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (3)$$

El resultado es un valor que representa la importancia del término t en el documento d relativo al corpus D .

Ejemplo 1. Aplicación de la métrica TF-IDF.

Considere el siguiente ejemplo numérico ilustrativo, basado en la formulación clásica de $TF-IDF$ descrita en [35]. Imaginemos que el término t aparece 20 veces en un documento que contiene un total de 100 palabras. La frecuencia de término (TF) para t se puede calcular de la siguiente forma:

$$TF = \frac{20}{100} = 0.2 \quad (4)$$

Supongamos ahora que una colección de documentos relacionados contiene un total de 10,000 documentos. Si el término t aparece en 100 de estos documentos, la frecuencia inversa de documento (IDF) se calcula como:

$$IDF = \log \left(\frac{10,000}{100} \right) = 2 \quad (5)$$

Utilizando estos dos valores, podemos calcular el puntaje $TF-IDF$ del término t para el documento:

$$TF-IDF = 0.2 \times 2 = 0.4 \quad (6)$$

Este cálculo muestra cómo *TF-IDF* asigna un peso a un término en un documento en función de su relevancia dentro del corpus.

Ejemplo 2. Cálculo de la importancia de una palabra dentro de un documento en relación con el corpus completo.

Suponga que se tiene el siguiente corpus compuesto por tres documentos cortos:

- Documento 1: “*La ironía en redes sociales es común*”
- Documento 2: “*El sarcasmo también puede ser ironía*”
- Documento 3: “*La literalidad y la ironía son opuestas*”

El término “*ironía*” aparece una vez en cada documento, mientras que la palabra “*la*” aparece en dos de ellos. Asumiendo que se omiten stopwords como “*la*”, y considerando que cada documento tiene el mismo número de palabras, el cálculo sería:

- $TF(ironía, d_1) = \frac{1}{6}$
- $IDF(ironía) = \log\left(\frac{3}{1+3}\right) = \log\left(\frac{3}{4}\right) \approx -0.1249$ (para este caso hipotético)
- $TF-IDF(ironía, d_1) \approx \frac{1}{6} \cdot (-0.1249) \approx -0.0208$

Este valor final representa la importancia de la palabra dentro de un documento en relación con el corpus completo. El valor obtenido puede formar parte de un vector de características que alimente a clasificadores como NEAT, máquinas de soporte vectorial (SVM, por sus siglas en inglés) o redes neuronales profundas. La principal fortaleza de *TF-IDF* radica en su capacidad para identificar términos distintivos en un documento al mismo tiempo que mitiga el impacto de palabras comunes. Esto lo hace ideal para aplicaciones donde es importante resaltar las características únicas de cada documento, como en motores de búsqueda y análisis de contenido. Sin embargo, como cualquier técnica, *TF-IDF* tiene sus limitaciones. Una de las más notables es la pérdida de contexto, ya que, al igual que BoW, ignora el orden y las relaciones semánticas entre las palabras. Además, puede ser susceptible a asignar pesos altos a términos poco frecuentes que no siempre son relevantes. Su dependencia del corpus también puede ser un desafío, especialmente en conjuntos de datos pequeños, donde la métrica *IDF* puede generar resultados inconsistentes.

4.4.3. BERT: *Bidirectional Encoder Representations from Transformers*

BERT es un modelo de red neuronal profunda basado en un mecanismo para redes neuronales llamado auto-atención (*self-attention*), el cual ha demostrado gran capacidad para modelar estructuras lingüísticas complejas. Este enfoque permite descubrir relaciones y secuencias dentro de la frase, lo que lo convierte en una herramienta especialmente adecuada para el procesamiento computacional del lenguaje natural [13].

En la primera oración, el término *bolsas* hace referencia a un objeto físico empleado para transportar productos; en la segunda, se utiliza con un sentido figurado, haciendo alusión a una manifestación física del cansancio. Esta ambigüedad semántica no puede resolverse adecuadamente si el modelo sólo analiza el contexto adyacente en una única dirección. Se estará cometiendo un error en al menos una de las dos oraciones dadas, si solo tratamos de predecir la naturaleza de la palabra “*bolsas*” tomando solo el contexto izquierdo o derecho. El modelo BERT aborda este problema mediante el mecanismo (*self-attention*), lo que le permite considerar simultáneamente tanto el contexto que precede como el que sigue a una palabra determinada.

4.5. Técnicas de reducción de características

Debido al gran tamaño vectorial de las representaciones textuales descritas previamente, es común aplicar técnicas de reducción de características que permitan disminuir la dimensionalidad sin perder información relevante. La reducción de características es una técnica fundamental en el procesamiento de datos de alta dimensionalidad, donde las representaciones textuales como BoW, BERT y TF-IDF generan vectores de gran tamaño. Estas representaciones suelen producir matrices dispersas y de alta dimensionalidad, lo que dificulta el entrenamiento eficiente de modelos de aprendizaje automático. Las técnicas de reducción de características buscan disminuir el número de variables manteniendo la mayor cantidad posible de información relevante, mejorando la eficiencia computacional y evitando problemas como el sobreajuste. En la siguiente sección se describe la técnica de Análisis de Componentes Principales, empleada en este trabajo como estrategia principal de reducción.

4.5.1. Análisis de Componentes Principales

El Análisis de Componentes Principales es una técnica estadística que transforma un conjunto de variables posiblemente correlacionadas en un nuevo conjunto de variables no correlacionadas llamadas componentes principales. Estos componentes se obtienen de manera que capturen la mayor cantidad posible de la varianza presente en los datos originales. La varianza mide cuánta dispersión presentan los datos respecto a su media. Por ejemplo, si se tienen múltiples observaciones de una misma variable, como la longitud de ciertas palabras en textos irónicos y no irónicos, una alta varianza implicaría que estas longitudes difieren considerablemente entre sí, mientras que una baja varianza indicaría una mayor homogeneidad. Esta noción de varianza es central para el ACP, ya que esta técnica busca identificar las direcciones (componentes) en el espacio de características a lo largo de las cuales los datos presentan la mayor variabilidad posible.

Definición Formal

Dado un conjunto de datos X con n observaciones y p variables, representado por una matriz $X \in \mathbb{R}^{n \times p}$, el objetivo del ACP es encontrar un nuevo conjunto de variables $Z = XW$, donde W es una matriz de coeficientes ortogonales que transforma las variables originales en componentes principales.

Cada componente principal Z_i es una combinación lineal de las variables originales, como se muestra en la ecuación (7):

$$Z_i = w_{i1}X_1 + w_{i2}X_2 + \dots + w_{ip}X_p \quad (7)$$

donde los coeficientes w_{ij} se obtienen al resolver un problema de optimización donde la varianza de las componentes principales es maximizada, sujeto a una restricción de ortogonalidad de la matriz de coeficientes W . Esta condición implica que los vectores que definen los componentes principales (las filas de la matriz de pesos W) deben ser ortogonales entre sí, es decir, deben ser linealmente independientes y mutuamente perpendiculares en el espacio vectorial. Esta propiedad asegura que la información capturada por un componente no se solape con la de los otros, garantizando así que cada componente principal aporte nueva información y no redundancia.

Cálculo del ACP

1. Estandarización de los datos:

Dado que las variables pueden tener diferentes escalas, es necesario estandarizarlas para que tengan media cero y varianza unitaria. La estandarización se realiza mediante la siguiente transformación:

$$X_{ij}^{\text{est}} = \frac{X_{ij} - \mu_j}{\sigma_j} \quad (8)$$

donde μ_j es la media de la variable j y σ_j su desviación estándar.

2. Cálculo de la matriz de covarianza:

La matriz de covarianza Σ de los datos estandarizados se calcula como:

$$\Sigma = \frac{1}{n-1} X^{\text{est}} (X^{\text{est}})^T \quad (9)$$

3. Obtención de eigenvalores y eigenvectores:

Se calculan los eigenvalores λ_i y los eigenvectores w_i de la matriz de covarianza Σ . Los eigenvalores representan la varianza explicada por cada componente principal, y los eigenvectores definen la dirección de los componentes.

4. Selección de los componentes principales:

Se seleccionan los k componentes principales que explican un porcentaje significativo de la varianza total. La proporción de varianza explicada por el i -ésimo componente se calcula como:

$$\text{Varianza explicada} = \frac{\lambda_i}{\sum_{j=1}^p \lambda_j} \quad (10)$$

El ACP permite proyectar los datos en un subespacio de menor dimensionalidad definido por los primeros k componentes principales, que retienen la mayor parte de la variabilidad de los datos originales. Esto facilita la visualización y el análisis de los datos, así como el entrenamiento de modelos de aprendizaje automático. En PLN, aplicar ACP a matrices generadas por BoW o TF-IDF permite reducir el número de términos relevantes, manteniendo la información esencial para tareas como clasificación de documentos y análisis de sentimientos [15, 36].

4.6. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) son una forma de emular, de una manera simplificada, el comportamiento del cerebro humano a través de modelos matemáticos implementados computacionalmente. Este tipo de herramientas, que forman parte del paradigma de aprendizaje supervisado, como técnica propia de la Inteligencia Artificial, funcionan recibiendo información a través de una capa de entrada, para posteriormente ser procesada por una serie de capas intermedias, y generar una clasificación a través de una capa de salida. Cada neurona, o *perceptrón*, va evolucionando a lo largo de un proceso de entrenamiento, permitiendo así mejorar la clasificación de los datos introducidos, pues los pesos de cada una de las sinapsis van siendo corregidos en función de si se llevó a cabo una clasificación correcta o incorrecta para cada ejemplo del conjunto de entrenamiento.

El aprendizaje profundo, por otro lado, consiste en la interconexión de diferentes RNA, cada una de éstas especializada en el entrenamiento de uno de los aspectos que llevan a la solución de un problema de clasificación, y permiten abordar, a su vez, la solución de problemas de clasificación tales como el reconocimiento de voz, la visión por computadora y el PLN. El desarrollo de este tipo de herramienta computacional se ha visto favorecido por la creciente disponibilidad de poder de cómputo.

La Figura 4 muestra la estructura de una RNA, y se puede observar que el elemento básico de su composición son las neuronas que se agrupan en varias de ellas y se establecen en capas, cada una constituyendo una funcionalidad propia.

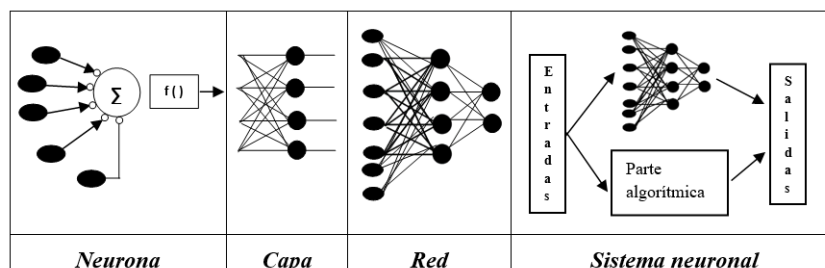


Figura 4: Estructura jerárquica de un sistema neuronal artificial. Tomada de [28].

En una RNA tradicional hay tres tipos de capas:

- **Capa de entrada (*Input layer*):** Esta primera capa está compuesta por neuronas que reciben las señales y datos del entorno, también se le conoce como capa sensorial.
- **Capa(s) oculta(s) (*Hidden layer(s)*):** Esta(s) capa(s) realiza(n) transformaciones no lineales de las entradas introducidas en la red. Cada capa oculta puede tener un número diferente de neuronas, que generalmente es mayor que el número de características. En sí, esta capa encuentra las representaciones internas correspondientes que ha recibido de la capa de entrada.
- **Capa de salida (*Output layer*):** La salida de la capa oculta se introduce en una función logística que convierte la salida de cada clase en una puntuación de probabilidad.

El diseño y la optimización de RNA's pueden requerir una búsqueda compleja en el espacio de arquitecturas posibles. En este contexto, surge un enfoque evolutivo: la **Neuroevolución**, y en particular, el algoritmo *NEAT (NeuroEvolution of Augmenting Topologies)*, que permite evolucionar tanto las topologías como los pesos de las redes neuronales. En la siguiente sección se profundiza en esta metodología.

4.7. Cómputo Evolutivo

El Cómputo Evolutivo (CE) está inspirado en los mecanismos de selección natural y evolución biológica. Esta disciplina emplea algoritmos basados en el principio darwiniano de *supervivencia del más apto* para resolver problemas de optimización y búsqueda [14]. Los métodos de computación evolutiva imitan procesos naturales como la reproducción, mutación, recombinación y selección para explorar el espacio de soluciones de un problema dado.

En el contexto de la computación evolutiva, el entorno natural se modela como un problema a resolver, mientras que los individuos representan soluciones candidatas. La aptitud (*fitness*) de cada individuo está determinada por qué tan bien resuelve el problema, lo que influye en su probabilidad de ser seleccionado y reproducirse en generaciones futuras.

La analogía entre la evolución natural y la resolución de problemas en el ámbito de la computación evolutiva, según lo planteado por Eiben y Smith [14], se establece de la siguiente manera:

Evolución	Resolución de Problemas
Entorno	Problema
Individuo	Solución Candidata
Aptitud (<i>fitness</i>)	Calidad de la solución

Tabla 2: Analogía entre la evolución natural y la resolución de problemas en computación evolutiva.

4.7.1. Algoritmos Genéticos

Los Algoritmos Genéticos (AG) son métodos de búsqueda heurística inspirados en el proceso de evolución biológica, propuestos inicialmente por Holland en la década de 1970 [23]. Un algoritmo genético busca encontrar la solución óptima de un problema a través de la evolución de una población de individuos. Cada individuo representa una posible solución al problema y se codifica mediante un conjunto de *genes*.

Sea $P(t)$ una población de N individuos en la generación t , donde cada individuo i se representa como un vector de genes $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$. El objetivo del algoritmo es maximizar (o minimizar) una función de aptitud $f(x_i)$, que evalúa la calidad de cada individuo en relación con el problema a resolver.

El proceso evolutivo se lleva a cabo en las siguientes etapas:

1. Inicialización:

Se genera aleatoriamente una población inicial de tamaño N .

2. Evaluación de aptitud:

Cada individuo x_i se evalúa mediante la función de aptitud $f(x_i)$, que determina su calidad como solución al problema.

3. Selección:

Se eligen individuos de la población actual $P(t)$ para formar una nueva población intermedia $P'(t)$, de acuerdo con un criterio de selección proporcional a la aptitud.

4. Cruza (recombinación):

Se seleccionan pares de individuos de $P'(t)$ y se combinan para generar nuevos individuos, llamados *hijos*, mediante un operador de cruza, donde se elige un punto aleatorio en la cadena de genes y se intercambian las subsecuencias.

5. Mutación:

Se aplica un operador de mutación a los individuos resultantes de la cruza, alterando aleatoriamente algunos de sus genes con una probabilidad p_m . La mutación introduce variabilidad genética y evita la convergencia prematura.

6. Reemplazo:

La nueva población $P(t + 1)$ se forma seleccionando los mejores individuos entre los hijos generados y, opcionalmente, algunos individuos de la población anterior, con la finalidad de mantener la diversidad.

En el Algoritmo 1 se muestra el pseudocódigo de un algoritmo genético estándar.

Algoritmo 1 Algoritmo Genético

- 1: Inicializar la población de tamaño N .
 - 2: **mientras** no se cumple el criterio de parada
 - 3: Evaluar la aptitud de cada individuo en $P(t)$.
 - 4: Seleccionar individuos para formar $P'(t)$.
 - 5: Aplicar cruza a los individuos seleccionados.
 - 6: Aplicar mutación a los individuos resultantes.
 - 7: Formar la nueva población $P(t + 1)$.
 - 8: $t \leftarrow t + 1$
 - 9: **fin mientras**
 - 10: Retornar la mejor solución encontrada.
-

4.7.2. Neuroevolución: NEAT (*NeuroEvolution of Augmenting Topologies*)

La Neuroevolución es un enfoque de aprendizaje automático que combina la evolución de redes neuronales con técnicas de computación evolutiva, inspirándose en la evolución del cerebro biológico en la naturaleza. Este enfoque ha demostrado ser especialmente eficaz en problemas donde la estructura óptima de la red no es conocida a priori o puede cambiar con el tiempo [39].

El algoritmo NEAT se diferencia de otros enfoques al evolucionar la estructura de la red y los pesos de sus conexiones de manera simultánea. Su objetivo es encontrar redes neuronales cada vez más complejas y efectivas, comenzando por estructuras simples y aumentando gradualmente su complejidad a lo largo de las generaciones.

NEAT se basa en los siguientes principios clave:

1. Representación directa de la topología:

NEAT utiliza un genoma directo para codificar tanto los nodos como las conexiones de una red neuronal. Cada genoma representa una red neuronal y contiene información sobre los nodos, conexiones, pesos y funciones de activación.

2. Especiación:

Para preservar la diversidad en la población y evitar que una única topología domine el proceso evolutivo, NEAT agrupa a los individuos en especies. Cada especie evoluciona por separado, lo que permite que las innovaciones estructurales se desarrollen sin ser rápidamente eliminadas por la selección natural.

3. Incremento gradual de la complejidad:

NEAT comienza con redes neuronales simples y, a lo largo de las generaciones, añade nodos y conexiones. Este enfoque, conocido como *complexification*, permite explorar un espacio de soluciones cada vez más amplio y sofisticado.

Funcionamiento de NEAT:

1. Inicialización de la población:

Se genera una población inicial de redes neuronales simples con pocas conexiones.

2. Evaluación de la aptitud:

Cada red neuronal se evalúa mediante una función de aptitud específica al problema que se desea resolver.

3. Especiación:

Se agrupan las redes en especies utilizando una métrica de compatibilidad que mide la diferencia entre genomas. Esto garantiza que las redes similares compitan entre sí y no con aquellas que tienen topologías muy diferentes.

4. Selección y reproducción:

Se seleccionan las redes más aptas dentro de cada especie para reproducirse. La reproducción puede incluir cruzamiento de genomas y mutación, donde se añaden nuevas conexiones o nodos a las redes existentes.

5. Reemplazo de la población:

La nueva generación se forma a partir de las redes resultantes del proceso de reproducción, y el ciclo se repite hasta que se alcanza un criterio de parada, como un número máximo de generaciones o una aptitud deseada.

En el Algoritmo 2 se muestra el pseudocódigo del algoritmo NEAT.

Algoritmo 2 Algoritmo NEAT

- 1: Inicializar la población de redes neuronales simples.
 - 2: **mientras** no se cumple el criterio de parada
 - 3: Evaluar la aptitud de cada red neuronal.
 - 4: Agrupar las redes en especies.
 - 5: Seleccionar las redes más aptas dentro de cada especie.
 - 6: Reproducir nuevas redes mediante cruza y mutación.
 - 7: Formar la nueva generación.
 - 8: **fin mientras**
 - 9: Retornar la mejor red neuronal encontrada.
-

A continuación, se describen los elementos fundamentales que permiten la aplicación del algoritmo NEAT dentro de este trabajo. Estos componentes —la codificación genética, los operadores genéticos, la especiación y la función de aptitud— constituyen la base del proceso evolutivo que guía la adaptación y mejora progresiva de las redes neuronales a lo largo de las generaciones. Su comprensión es esencial para entender cómo el modelo optimiza su desempeño a partir de poblaciones iniciales simples.

Codificación Genética y Operadores Genéticos. Cada red neuronal en NEAT está representada por un *genoma*, que incluye una lista de *genes de conexión*, cada uno con un *peso*, un *indicador de habilitación* que puede activar o desactivar la conexión, y dos *genes de nodo* que especifican la conexión de entrada y salida. La codificación genética se expresa mediante la siguiente estructura:

$$G = (N, C) \tag{11}$$

donde G representa el genoma, N es el conjunto de nodos, y C el conjunto de conexiones.

Los operadores genéticos incluyen:

- **Cruza:** Combina dos genomas para crear un descendiente, tomando en cuenta la compatibilidad de los genes mediante el cálculo de la distancia genética:

$$d = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{12}$$

donde E y D son el número de genes excedentes y disjuntos respectivamente, N es el número total de genes en el genoma más grande, $\overline{W}$ es la diferencia media de pesos coincidentes, y c_1 , c_2 , c_3 son coeficientes que ajustan la importancia de estos factores.

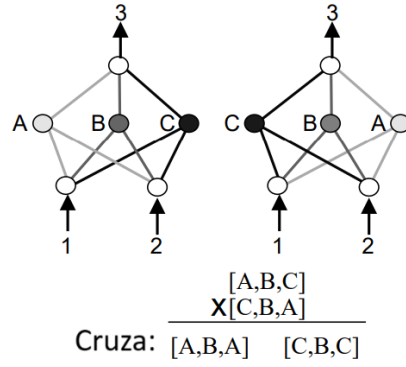


Figura 5: Ejemplo de cruce en NEAT. Adaptado de [39].

- **Mutación:** Altera los pesos de las conexiones o añade nuevos nodos y conexiones. La mutación de peso se modela comúnmente como una perturbación gaussiana:

$$w' = w + N(0, \sigma^2) \quad (13)$$

donde w es el peso original, w' es el peso mutado, y $N(0, \sigma^2)$ es una variable aleatoria normal con media 0 y varianza σ^2 .

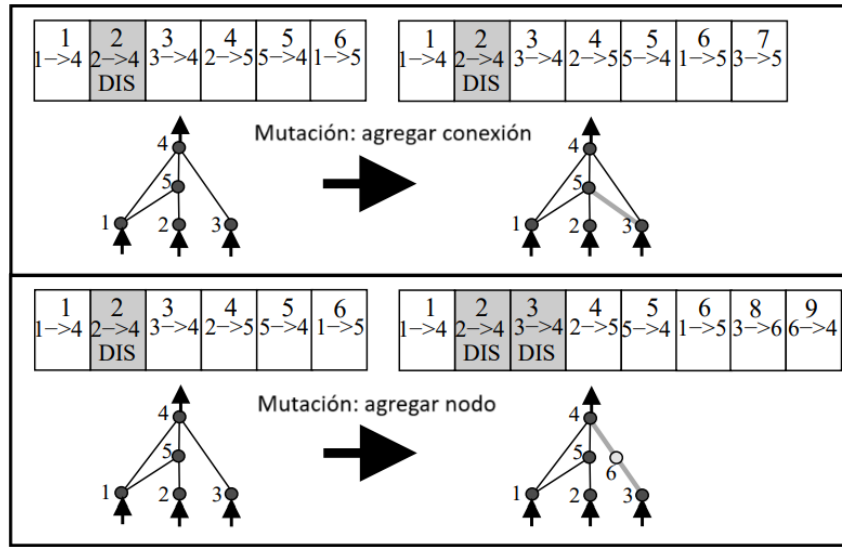


Figura 6: Ejemplo de mutación en NEAT: adición de conexión y de nodo. Adaptado de [39].

Especiación y Función de Aptitud. La especiación protege las innovaciones genéticas permitiendo que genomas similares compitan entre sí, lo que reduce el riesgo de que innovaciones beneficiosas sean descartadas prematuramente. Los genomas se agrupan en especies basados en la distancia genética d , utilizando un umbral δ para determinar si un genoma pertenece a una especie existente.

La *función de aptitud* f para la detección de ironía se diseña para evaluar qué tan bien una red neuronal identifica y clasifica correctamente los *tweets* irónicos. Se puede

expresar como una combinación del accuracy y el *recall* (proporción de verdaderos positivos detectados respecto al total real de positivos), por un factor que penaliza las redes excesivamente complejas para fomentar la simplicidad estructural:

$$f = \alpha(\text{Precision}) + \beta(\text{Recall}) - \gamma(|N| + |C|) \quad (14)$$

donde α , β , y γ son coeficientes que equilibran la precision, el recall, y la penalización por complejidad, respectivamente.

5. Metodología

En este capítulo se describen los procedimientos metodológicos empleados para el desarrollo del presente trabajo de investigación. Se detallan las características del corpus utilizado, el preprocesamiento de datos, las técnicas de representación textual aplicadas —como BoW, TF-IDF, reducción de dimensionalidad mediante ACP y los *embeddings* contextuales de BERT—, así como los algoritmos de clasificación implementados, específicamente NEAT y SVM. Además, se explica el diseño experimental, las métricas de evaluación empleadas y las pruebas estadísticas realizadas para validar los resultados. Esta sección tiene como propósito ofrecer una visión integral del flujo metodológico seguido, desde la preparación de los datos hasta el análisis comparativo de los modelos.

5.1. Datos

Los datos empleados en este trabajo provienen del repositorio de la Tarea 3 de SemEval-2018 [42], específicamente dedicada a la detección de ironía en *tweets* en Inglés. En particular, se emplea la Tarea A, que es una clasificación binaria, donde las etiquetas están definidas como 0 (no irónico) y 1 (irónico). La estructura del dataset consiste en un archivo separado por tabulaciones, con una línea correspondiente a cada *tweet*. Cada línea incluye el índice del *tweet*, una etiqueta de clasificación binaria y el texto del *tweet*.

La recopilación de los *tweets* se llevó a cabo utilizando la API de Twitter, seleccionando mensajes que contenían los hashtags *#irony*, *#sarcasm* y *#not*, lo que permitió obtener un total de 4,792 *tweets*. El etiquetado de los *tweets* se realizó de manera manual, siguiendo el esquema propuesto por Van Hee et al. en [42], garantizando así la calidad y coherencia de las etiquetas asignadas. Finalmente, el corpus utilizado consta de 3,834 *tweets* para el conjunto de entrenamiento y 784 para el conjunto de prueba. La Tabla 3 muestra ejemplos de *tweets* utilizados en el estudio, junto con sus respectivas etiquetas.

Tweet	Etiqueta
<i>Yay for another work at 4am day :neutral_face:</i>	1 (irónico)
<i>Had a nice 3 hour nap. Picking up my Chinese take-out and set to watch #GuardiansOfTheGalaxy</i>	0 (no irónico)
<i>Sweet United Nations video. Just in time for Christmas. #imagine #NoReligion http://t.co/fej2v3OUBR</i>	0 (no irónico)
<i>You're never too old for Footie Pajamas.</i>	0 (no irónico)
<i>Just great when you're mobile bill arrives by text</i>	1 (irónico)

Tabla 3: Ejemplo de *tweets* etiquetados

5.2. Preprocesamiento de datos

Este corpus ya incluye un procesamiento preliminar significativo para asegurar la calidad de los datos:

- **Limpieza de datos:** Se eliminaron retweets (republicación de un mensaje previamente emitido por otro usuario en la plataforma), duplicados y *tweets*

que no estaban en Inglés.

- **Normalización de texto:** Se reemplazaron los caracteres con escape XML y se convirtieron los emojis a descripciones en UTF-8. Adicionalmente, se limpiaron los hashtags relacionados con la ironía para evitar sesgos en la clasificación.
- **Remoción de elementos no textuales:** Se eliminaron enlaces, menciones, espacios en blanco y se removió el símbolo de hashtag (#).
- **Conversión de texto:** Todas las letras mayúsculas fueron convertidas a minúsculas para mantener la consistencia en el análisis.
- **Tokenización del *tweet*:** El texto se divide en unidades mínimas llamadas *tokens*. Un *token* es una palabra. Este proceso permite descomponer el lenguaje en elementos discretos para ser analizados individualmente.
- **Eliminación de stopwords, palabras manualmente especificadas y con menos de 5 apariciones:** Se removieron palabras especificadas que no aportaban valor al análisis con el fin de reducir el ruido y mejorar la calidad de las representaciones textuales.

La Tabla 4 muestra un ejemplo de un *tweet* original y su versión preprocesada, reduciéndolo a un conjunto de palabras clave relevantes.

Tabla 4: Ejemplo de tweet original y preprocesado.

Tweet original	Tweet preprocesado
<i>Sweet United Nations video. Just in time for Christmas. #imagine #NoReligion</i> <i>http://t.co/fej2v3OUBR</i>	sweet video time christmas

Limpieza adicional y refinamiento del corpus

Después del preprocesamiento inicial, el corpus estaba compuesto por un total de 4,007 *tweets*. Con el objetivo de mejorar la calidad del conjunto de datos y reducir la presencia de ruido, se implementaron pasos adicionales de limpieza.

Estas etapas incluyeron:

- **Eliminación de tokens no informativos**, como palabras no reconocibles o construcciones que no forman parte del lenguaje natural. Ejemplo: @user1234!!
- **Normalización de repeticiones excesivas de letras.** Ejemplo: buenoooooooo → bueno.
- **Eliminación de palabras hápax**, es decir, aquellas que solo aparecieron una vez en todo el corpus.

Como resultado de esta depuración avanzada, el corpus final quedó conformado por 4,324 *tweets*, ya que se reincorporaron algunos ejemplos previamente descartados al cumplir ahora con los criterios mínimos tras la limpieza. A partir de este conjunto

depurado, se generó una representación binaria basada en el modelo de BoW, compuesta por 2,860 términos únicos. Esta representación refinada fue posteriormente utilizada como entrada para la técnica de reducción de dimensionalidad mediante ACP, produciendo una versión comprimida del corpus con 50 componentes principales. Dicha representación reducida sirvió como base para los experimentos con los modelos clasificadores NEAT y SVM.

5.3. Transformación de los datos con *Bag of Words* Binaria

El modelo de BoW Binaria se basa en la creación de una matriz donde las filas representan los *tweets* y las columnas corresponden a las palabras únicas (vocabulario) extraídas del corpus después del preprocesamiento. Los valores de la matriz son binarios:

- **0:** Indica que la palabra no está presente en el *tweet*.
- **1:** Indica que la palabra está presente en el *tweet*.

La Tabla 5 muestra un ejemplo de la matriz generada tras aplicar el modelo de BoW al corpus preprocesado. En esta etapa aún no se ha aplicado la limpieza adicional ni el refinamiento posterior, por lo que se conservan todas las palabras que superaron las etapas básicas de normalización. La matriz resultante corresponde a un total de 4,007 *tweets* y 846 palabras únicas.

Tabla 5: Ejemplo de matriz de *Bag of Words* Binaria resultante.

absolutely	action	adorable	agree	alive	album	amazing	angry
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0

La matriz presentada es solo un fragmento del total de datos transformados. Debido a la alta dimensionalidad de la matriz, no es posible realizar una representación gráfica en dimensiones mayores a tres.

Este proceso transforma cada *tweet* en un vector de características de longitud fija que representa la presencia o ausencia de palabras clave, capturando así la información relevante del texto. Tras aplicar el modelo de BoW Binaria y generar la correspondiente matriz, debido a su elevada dimensionalidad, se procede a reducirla mediante ACP. La matriz resultante, con una dimensionalidad reducida y más manejable, se utiliza posteriormente como entrada para el algoritmo NEAT.

5.4. Reducción con ACP

Dado que la matriz generada mediante el modelo de BoW Binaria posee una alta dimensionalidad. La finalidad de esta reducción es optimizar los recursos computacionales y mejorar el rendimiento del algoritmo NEAT al reducir el número de características, manteniendo al mismo tiempo la mayor cantidad posible de información relevante.

El ACP busca transformar la matriz original $X \in \mathbb{R}^{n \times p}$, donde $n = 4007$ es el número de *tweets* y $p = 846$ es el número de palabras únicas, en una nueva matriz $Z \in \mathbb{R}^{n \times k}$, donde $k \ll p$ representa el número de componentes principales seleccionados. Esta transformación se realiza mediante la siguiente expresión:

$$Z = XW \quad (15)$$

donde $W \in \mathbb{R}^{p \times k}$ es la matriz cuyas columnas corresponden a los eigenvectores asociados a los k eigenvalores más grandes de la matriz de covarianza de X . Cada eigenvalor λ_i representa la varianza explicada por el componente principal correspondiente, y los componentes se ordenan de manera decreciente según la cantidad de varianza que explican.

Para seleccionar el número óptimo de componentes principales, se llevaron a cabo experimentos utilizando diferentes valores de k , evaluando el impacto de cada selección en el rendimiento del modelo. En este contexto, la varianza explicada acumulada constituye una métrica esencial, ya que permite cuantificar cuánta información del conjunto original se preserva al proyectar los datos en un espacio de menor dimensión. Esta medida se calcula mediante la siguiente expresión:

$$\text{Varianza explicada acumulada}(k) = \frac{\sum_{i=1}^k \lambda_i}{\sum_{j=1}^p \lambda_j} \quad (16)$$

donde el numerador representa la suma de los k eigenvalores más grandes (ordenados de mayor a menor) y el denominador es la suma total de todos los eigenvalores de la matriz de covarianza. Esta proporción permite determinar cuántos componentes principales son necesarios para retener un porcentaje deseado de la información total.

La selección final de k se basó en el porcentaje de varianza explicada deseado, manteniendo un equilibrio entre la reducción de dimensionalidad y la preservación de la información relevante. Los valores específicos de k probados, así como el análisis de los resultados obtenidos, se detallarán en el capítulo de **Experimentación y Resultados**. La matriz reducida Z , generada tras aplicar el ACP, se utilizó como entrada para el algoritmo NEAT. Al reducir significativamente el número de características, se logró disminuir el tiempo de entrenamiento del modelo debido a su menor complejidad, y mejorar su capacidad de generalización, evitando problemas derivados de la alta dimensionalidad original de los datos.

5.4.1. Selección de Componentes para obtener el 70 % de Varianza Explicada

Para reducir la dimensionalidad de los datos de entrada manteniendo la mayor cantidad de información posible, se determina el número mínimo de componentes necesarios para explicar al menos el 70 % de la varianza total del conjunto de datos.

El procedimiento aplicado se detalla a continuación:

1. Cálculo de la varianza explicada acumulada:

Se calcularon los eigenvalores y eigenvectores de la matriz de covarianza, obteniendo la varianza explicada por cada componente principal. Posteriormente, se calculó la varianza explicada acumulada.

2. Selección de componentes:

Se seleccionaron los primeros 278 componentes principales, que explican aproximadamente el 70 % de la varianza total, como se muestra en la Figura 7.

3. Reconstrucción de datos:

Se reconstruyó la matriz de datos utilizando únicamente los 278 componentes seleccionados, generando una representación de menor dimensionalidad que conserva la mayor parte de la información relevante.

4. Obtención de coeficientes de los componentes:

Se extrajeron los coeficientes de los 278 componentes principales seleccionados, que indican la contribución de cada palabra a la formación de los nuevos componentes.

5. Cálculo de la norma de cada palabra:

Para evaluar la importancia de cada palabra en los componentes seleccionados, se calculó la norma de los coeficientes correspondientes a cada palabra.

6. Ordenamiento y selección de palabras:

Se ordenaron las palabras de acuerdo a su norma, seleccionando las 335 palabras con mayor norma, que fueron consideradas como las más representativas del conjunto de datos.

7. Creación de *Bag of Words* Reducida:

Finalmente, se generó una nueva matriz de datos utilizando únicamente las 335 palabras seleccionadas, reduciendo así las dimensiones originales de (4007, 846) a (4007, 335).

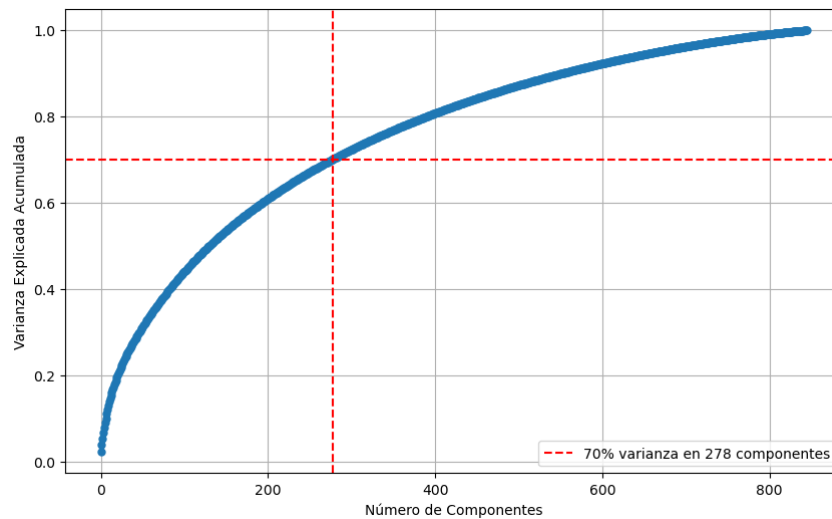


Figura 7: Varianza explicada acumulada por componentes principales

5.5. Algoritmo Genético: Parámetros generales de NEAT

Para la detección de ironía en *tweets*, se adaptó el algoritmo NEAT. La configuración específica del algoritmo se detalla a continuación. Esta configuración se definió

tomando como referencia recomendaciones metodológicas reportadas en la literatura [39]. En particular, se privilegiaron parámetros que favorecen la exploración de arquitecturas iniciales simples con una complejidad creciente controlada, lo cual permite mantener un equilibrio entre capacidad de generalización y costo computacional.

Población inicial. Consistió en 20 genomas, con el criterio de aptitud configurado para maximizar y un umbral de aptitud establecido en 1, lo que indica la búsqueda de una solución perfecta. La configuración también especifica que no se reinicie la población en caso de extinción.

Configuración del Genoma. Los nodos de la red fueron inicialmente configurados para utilizar una combinación de funciones de activación sigmoideal y tangente hiperbólica, seleccionadas aleatoriamente. Los pesos de las conexiones se inicializan con una media de 0.0 y una desviación estándar de 1.0, con valores que pueden oscilar entre -5 y 5. Esta configuración busca garantizar una distribución equilibrada de valores positivos y negativos al comienzo del proceso evolutivo. Las mutaciones en los pesos y los sesgos se establecieron con altas tasas para permitir una exploración eficiente del espacio de soluciones.

Topología de la Red. La red es de tipo feed-forward con conexiones iniciales completas entre las capas de entrada y salida. Se especificó un número de entradas, correspondientes a las dimensiones de la matriz generada por BoW y reducida por ACP (de acuerdo al experimento), y 2 salidas, representando las clases irónico y no irónico. Las probabilidades de adición y eliminación de conexiones se establecieron en 0.5 para fomentar la evolución dinámica de la topología.

Especiación y Reproducción. Se utiliza un umbral de compatibilidad de 3.0 para determinar la especiación (este umbral define cuán similares deben ser dos genomas para pertenecer a la misma especie dentro del algoritmo NEAT), con un máximo de 20 generaciones de estancamiento antes de que una especie sea considerada estancada. La reproducción está configurada para enfatizar el elitismo y la supervivencia de los individuos más aptos, asegurando que las características deseables se mantengan a través de las generaciones.

La Tabla 6 resume la configuración de los parámetros utilizados para el algoritmo NEAT en este estudio. Estos parámetros fueron seleccionados mediante pruebas preliminares con el fin de optimizar el rendimiento del modelo en la tarea de detección de ironía en *tweets*.

Parámetros Generales	
Población inicial	20
Función de aptitud	Maximización, umbral de aptitud = 1
Extinción	La población no se resetea
Nº de generaciones	500
Configuración del genoma	
Funciones de activación	Sigmoidal y tangente hiperbólica, seleccionadas aleatoriamente
Pesos de las conexiones (determinan la fuerza y dirección)	Media = 0, Std = 1, Rango = [-5, 5]
Mutación	40 %
Topología de la Red	
Tipo	Feed-forward
Conexiones iniciales	Completas entre capas de entrada y salida
Entradas	De acuerdo al experimento
Salidas	2 (irónico, no irónico)
Reproducción	Elitismo y supervivencia de los individuos más aptos

Tabla 6: Configuración del algoritmo NEAT

5.6. Procedimiento experimental con NEAT

Para integrar todos los componentes descritos, se diseñó un flujo experimental que guiara la ejecución de los experimentos. El diagrama de la Figura 8 resume el proceso seguido en esta investigación.

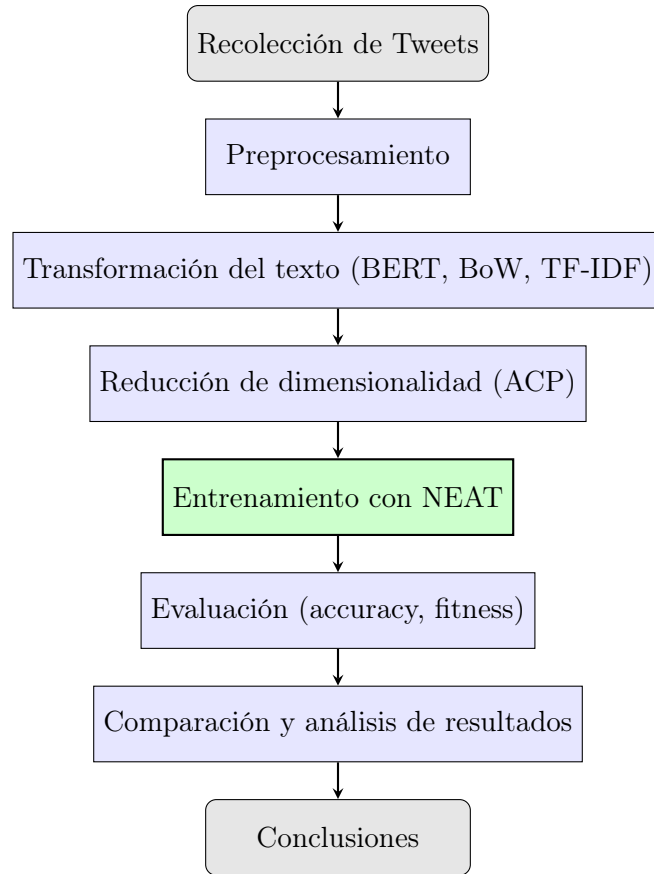


Figura 8: Flujo del proceso experimental con NEAT

Para complementar el flujo general mostrado anteriormente, en la Figura 9 se presenta el proceso interno del entrenamiento con NEAT, destacando las etapas que conforman su ciclo evolutivo.

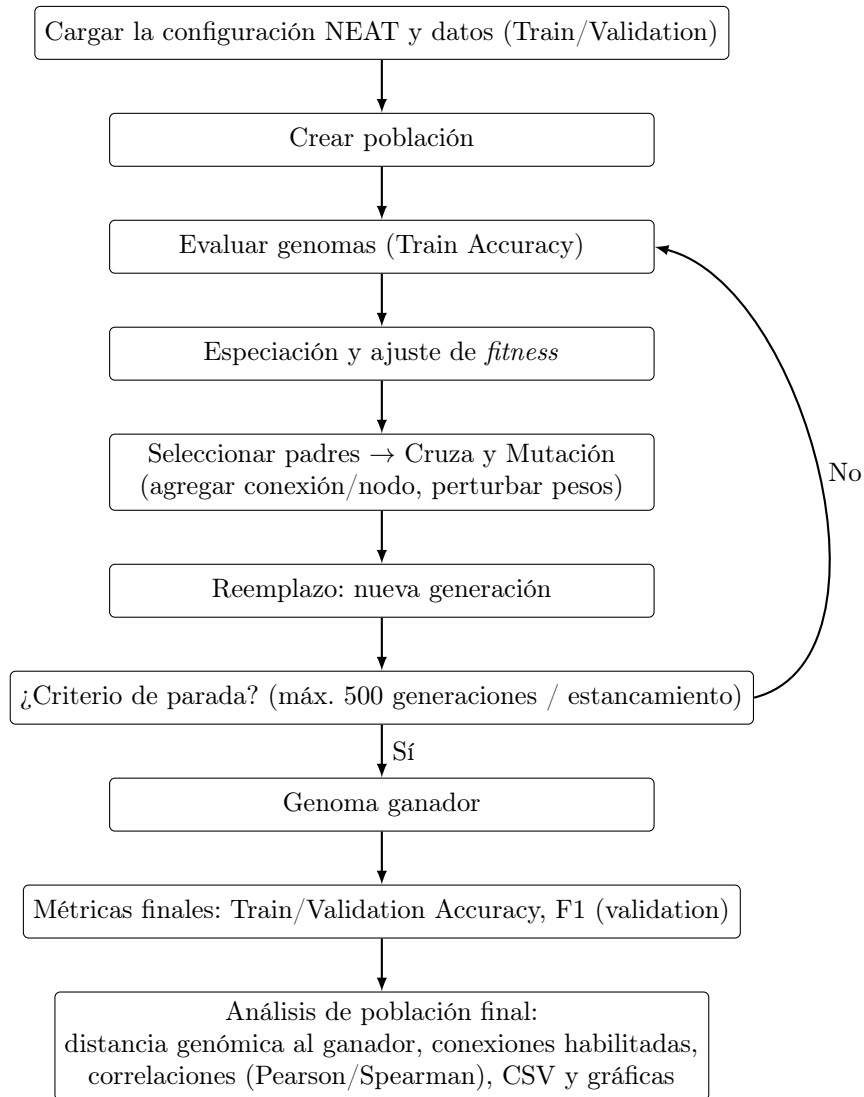


Figura 9: Operación interna de NEAT: el bucle evolutivo itera evaluación → especiación/selección → cruza y mutación → reemplazo hasta el criterio de parada; al final se calculan *validation accuracy*, F1 y se realiza el análisis de genomas.

6. Experimentación y Resultados

6.1. Experimentos

En esta sección se describen los experimentos realizados para evaluar el desempeño del algoritmo NEAT y de un clasificador SVM, empleando diversas representaciones de texto y niveles de reducción de características mediante ACP. Todos los experimentos fueron aplicados con ambos algoritmos (NEAT y SVM); sin embargo, NEAT fue evaluado con 15 ejecuciones independientes. Este número de ejecuciones fue definido como un compromiso entre la significancia estadística de los resultados y la factibilidad computacional, dado el alto costo de entrenamiento que implica la evolución de redes neuronales desde cero. Mientras que, SVM fue ejecutado únicamente una vez por configuración, ya que su entrenamiento no depende de poblaciones evolutivas ni variabilidad estocástica. Además, se eligió SVM como método de comparación debido a su eficacia comprobada en tareas de clasificación de texto y su uso extendido como línea base en problemas de PLN, mostrando buena capacidad de generalización y estabilidad incluso con grandes volúmenes de características [20, 24], lo que permite contrastar el desempeño de NEAT frente a un modelo supervisado robusto y ampliamente validado en la literatura.

En el caso de NEAT, se exploraron cuatro niveles de reducción dimensional mediante ACP: 50, 100, 200 y 278 componentes principales. Estas reducciones se aplicaron sobre la representación *Bag of Words* Binaria original de 846 dimensiones. Adicionalmente, se evaluó el rendimiento de NEAT con una representación BoW reducida manualmente a 335 características, obtenida a través de un filtrado por frecuencia mínima, así como una versión mejorada del corpus, producto de una limpieza profunda y una reducción posterior a 50 componentes principales mediante ACP (BoW clean + ACP 50).

Tanto NEAT como SVM fueron también evaluados con representaciones obtenidas a partir de BERT. Estos vectores se utilizaron como entrada sin ningún tipo de procesamiento textual adicional, con el fin de conservar la riqueza contextual propia de este modelo de lenguaje. Todos los experimentos fueron realizados bajo condiciones equivalentes para garantizar una comparación justa y consistente entre representaciones, configuraciones y algoritmos.

6.2. Entorno y herramientas a emplear

El proyecto es implementado en el lenguaje de programación Python. Para la exploración, transformación y modelado de los datos, se emplearon módulos como Matplotlib y Seaborn para la visualización, así como sklearn para la extracción de características (CountVectorizer, TfidfVectorizer), y reducción dimensional.

Todos los experimentos computacionales fueron realizados en un equipo ASUS TUF Gaming F15, equipado con un procesador Intel Core i5-12500H de 12^a generación a 2.50 GHz, 16 GB de memoria RAM y sistema operativo Windows 11 de 64 bits.

6.3. Resultados

En las Tablas 8, 9, 10 y 11 se muestran los resúmenes estadísticos obtenidos tras aplicar NEAT utilizando una representación de datos reducida mediante ACP. Se reportan las métricas de promedio, mediana, desviación estándar, valor mínimo y valor máximo para *fitness*, *train accuracy* —que corresponde al desempeño del modelo sobre los datos utilizados en el entrenamiento— y *validation accuracy* —que mide la capacidad del modelo para generalizar a un conjunto de datos no visto durante el entrenamiento— a lo largo de las 15 ejecuciones realizadas. En este trabajo, el *fitness* y la *train accuracy* presentan valores idénticos debido a que la función de evaluación de NEAT se definió directamente como la proporción de aciertos sobre el conjunto de entrenamiento. Durante la evolución, el *fitness* de cada genoma se calcula incrementando un contador cada vez que la red clasifica correctamente un *tweet* y dividiendo este valor entre el número total de ejemplos de entrenamiento, lo que equivale conceptualmente a la *train accuracy*. Aun así, se reportan ambas métricas de manera explícita: el *fitness* representa el criterio de selección evolutiva que guía el proceso de optimización, mientras que la *train accuracy* se incluye como métrica de desempeño sobre el conjunto de entrenamiento.

Cada ejecución experimental con el modelo NEAT genera un archivo denominado `predicciones_neat.csv`, el cual contiene las columnas Índice (identificador único del tweet en el corpus), Etiqueta verdadera y Etiqueta predicha por el modelo. Este archivo permite analizar de manera puntual el desempeño del modelo sobre instancias específicas. En la Tabla 7 se presentan algunos ejemplos extraídos de dicho archivo. Estos casos muestran situaciones de predicciones correctas y errores, lo que facilita interpretar las métricas cuantitativas obtenidas en los experimentos.

Índice	Tweet (preprocesado)	Etiqueta verdadera	Etiqueta predicha
0	yay for another work at 4am day neutral face	1	1
1	picking up my chinese take out and set to watch	0	0
2	just great when your mobile bill arrives by text	1	1
3	i love how everything is so easy today	1	0
4	just finished watching guardians of the galaxy	0	0

Tabla 7: Ejemplos de predicciones obtenidas por el modelo NEAT a partir del archivo `predicciones_neat.csv`.

A continuación, los Gráficos 10, 11, 12 y 13 ilustran el proceso de convergencia del mejor *fitness* a lo largo de las 500 generaciones del algoritmo NEAT, utilizando la representación de datos reducida mediante ACP. Se muestra el *fitness* promedio

de la población (línea azul), así como el mejor *fitness* alcanzado en cada generación (línea roja). Las líneas punteadas representan una desviación estándar por encima y por debajo del *fitness* promedio.

6.3.1. Resultados de NEAT aplicando ACP

a. Con 50 componentes

El experimento con **50 componentes principales** muestra un rendimiento consistente, logrando un *validation accuracy* promedio de 0.6024 con una baja desviación estándar de 0.0122, lo que indica estabilidad en las ejecuciones (ver Tabla 8). La convergencia del mejor *fitness* se alcanzó rápidamente durante las primeras 100 generaciones, estabilizándose cerca de 0.60 y mostrando una variabilidad moderada controlada, posteriormente experimenta pequeñas fluctuaciones hasta estabilizarse.

Las líneas verdes punteadas, que representan una desviación estándar por encima y por debajo del *fitness* promedio, muestran que existe una variabilidad moderada en la población durante el proceso evolutivo (ver Figura 10). Sin embargo, esta variabilidad se mantiene dentro de límites controlados, lo que sugiere que el algoritmo es capaz de explorar de manera efectiva el espacio de soluciones sin caer en una convergencia prematura.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.6064	0.6064	0.6024
Mediana	0.6062	0.6062	0.6061
Desviación Estándar	0.0042	0.0042	0.0122
Min	0.6	0.6	0.5669
Max	0.6146	0.6146	0.6162

Tabla 8: Resultados estadísticos de NEAT aplicando ACP con 50 componentes

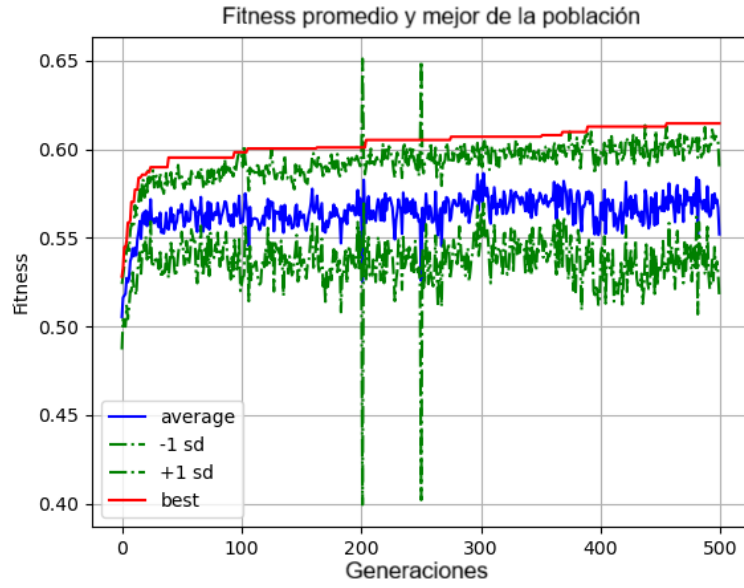


Figura 10: Gráfico de convergencia del mejor *fitness* con ACP (50 componentes)

b. Con 100 componentes

El experimento con 100 componentes principales mostró un mínimo decremento respecto a la configuración con 50 componentes, alcanzando un accuracy promedio de validación y una mediana de 0.5988, con una desviación estándar de 0.0219, lo que indica estabilidad en las ejecuciones (ver Tabla 9). La convergencia del mejor *fitness* se logró de manera gradual y estable durante las 500 generaciones, estabilizándose alrededor de 0.61, con una variabilidad controlada en la población (ver Figura 11). Estos resultados evidencian que aumentar el número de componentes aporta información adicional relevante que mejora el rendimiento, sin incrementar de manera significativa la variabilidad.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.6092	0.6092	0.5988
Mediana	0.6092	0.6092	0.5988
Desviación Estándar	0.0053	0.0053	0.0219
Min	0.598	0.598	0.5406
Max	0.6173	0.6173	0.625

Tabla 9: Resultados estadísticos de NEAT aplicando ACP con 100 componentes

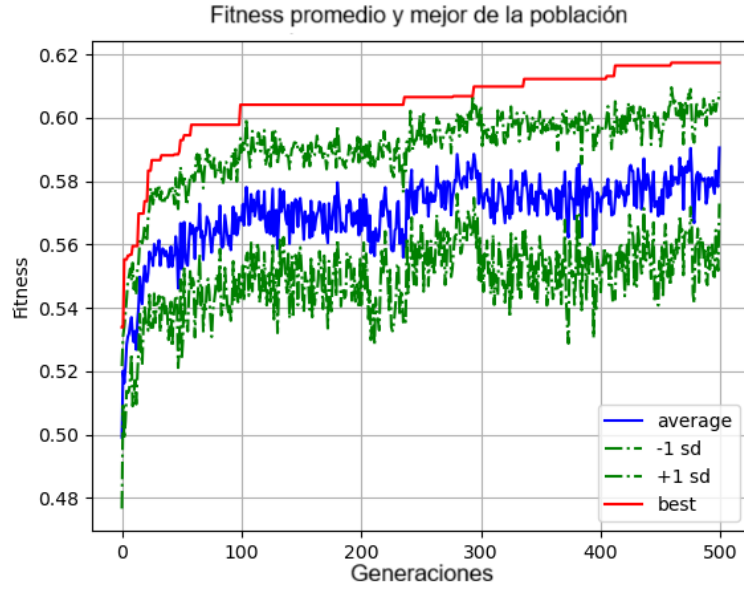


Figura 11: Gráfico de convergencia del mejor *fitness* con ACP (100 componentes)

c. Con 200 componentes

El experimento con 200 componentes principales muestra un rendimiento similar al de configuraciones con menor cantidad de componentes, obteniendo un *validation accuracy* promedio de 0.5809 y una baja desviación estándar de 0.0149, lo que refleja una buena estabilidad en los resultados. Los valores extremos de *validation accuracy* se situaron entre 0.5509 y 0.6105, destacando que el modelo mantuvo un buen desempeño sin grandes fluctuaciones (ver Tabla 9). El gráfico de convergencia muestra que el mejor *fitness* se estabiliza en torno a 0.61 después de las primeras 100 generaciones, manteniendo una variabilidad moderada controlada en la población durante el resto del proceso evolutivo (ver Figura 12). Estos resultados indican que, aunque aumentar el número de componentes a 200 no aporta una mejora significativa, sí permite mantener un desempeño competitivo con una representación más detallada de los datos.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.6122	0.6122	0.5809
Mediana	0.6116	0.6116	0.5843
Desviación Estándar	0.0044	0.0044	0.0149
Min	0.6044	0.6044	0.5509
Max	0.6225	0.6225	0.6105

Tabla 10: Resultados estadísticos de NEAT aplicando ACP con 200 componentes

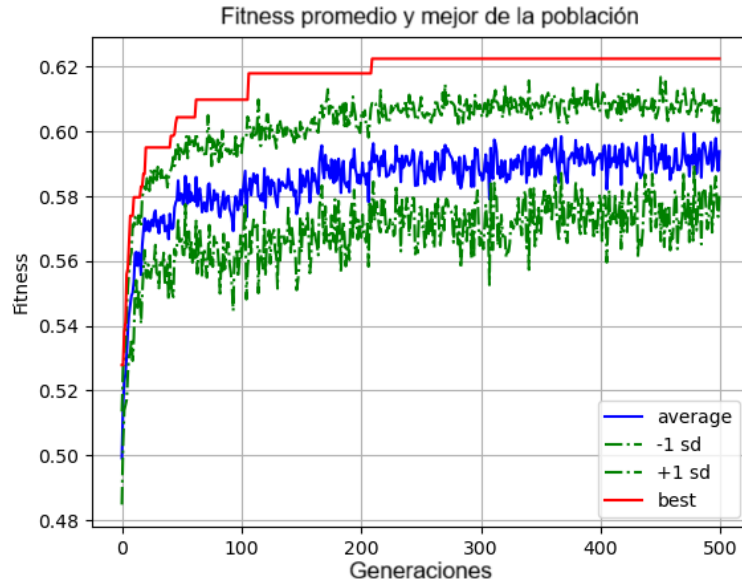


Figura 12: Gráfico de convergencia del mejor *fitness* con ACP (200 componentes)

d. Con 278 componentes (70 % de varianza explicada)

El experimento con 278 componentes principales, correspondiente al 70 % de varianza explicada, presenta un rendimiento ligeramente inferior en comparación con configuraciones de menor cantidad de componentes, con un *validation accuracy* promedio de 0.5653. La desviación estándar indica una variabilidad ligeramente mayor respecto a los experimentos con 100 y 200 componentes, lo que sugiere que aumentar el número de componentes no necesariamente contribuye a una mayor estabilidad del modelo. Los valores extremos de *validation accuracy* oscilaron entre 0.5276 y 0.6017, lo que refleja un rango de desempeño más amplio (ver Tabla 11).

En cuanto al proceso de convergencia, el mejor *fitness* se estabilizó en torno a 0.61 después de las primeras 100 generaciones, similar a lo observado en configuraciones previas. No obstante, el gráfico de convergencia muestra una mayor fluctuación en el *fitness* promedio de la población, lo que sugiere que la mayor cantidad de componentes introdujo una complejidad adicional que dificultó una convergencia más estable (ver Figura 13). Aunque la configuración con 278 componentes garantiza que se retenga el 70 % de la varianza del conjunto de datos original, el rendimiento obtenido no justifica el aumento en la dimensionalidad y el costo computacional asociado. Configuraciones con menos componentes, como 100 o 200, ofrecen un mejor equilibrio entre *accuracy* y eficiencia.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.6104	0.6104	0.5653
Mediana	0.6116	0.6116	0.5640
Desviación Estándar	0.0054	0.0054	0.0264
Min	0.6008	0.6008	0.5276
Max	0.6204	0.6204	0.6017

Tabla 11: Resultados estadísticos de NEAT aplicando ACP con 278 componentes

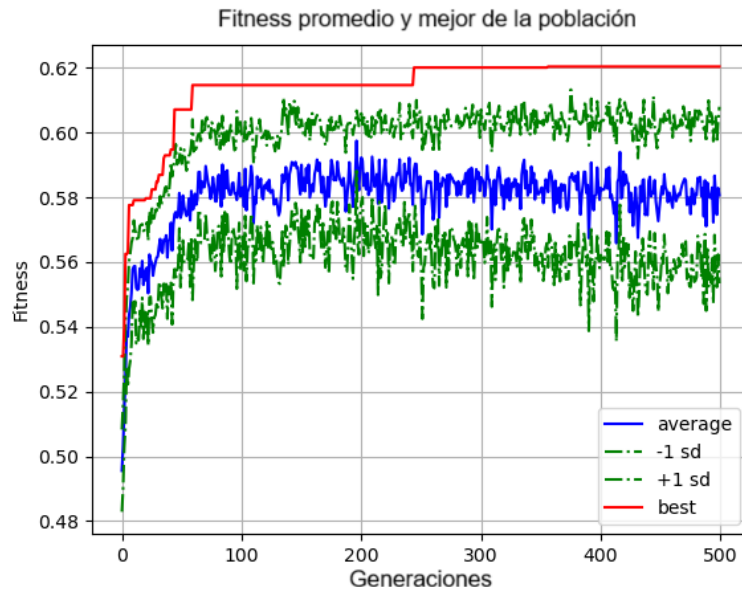


Figura 13: Gráfico de convergencia del mejor *fitness* con ACP (278 componentes)

6.3.2. Resultados de NEAT con *Bag of Words* Binaria

a. Reducida

El experimento utilizando BoW Binaria reducida a 335 palabras muestra que el modelo es capaz de alcanzar resultados competitivos. Los resultados estadísticos obtenidos se muestran en la Tabla 12. El *validation accuracy* promedio alcanzó un valor de 0.5641, con una desviación estándar relativamente baja (0.0303), lo cual sugiere estabilidad en el comportamiento del modelo a lo largo de las ejecuciones. Asimismo, el mejor desempeño en validación alcanzó un 0.6061, evidenciando que, aunque la representación BoW es más simple, permite al algoritmo alcanzar niveles de generalización razonables. El gráfico de convergencia (ver Figura 14) muestra que el mejor *fitness* se estabiliza alrededor de 0.58 después de las primeras 150 generaciones, con fluctuaciones más pronunciadas en comparación con las configuraciones de ACP. Esto sugiere que la representación con BoW reducida introduce una mayor complejidad en el espacio de búsqueda evolutivo, lo que genera una convergencia menos estable. Aunque esta configuración es menos eficiente en términos de estabilidad y presenta una mayor variabilidad, ofrece una solución alternativa viable al reducir

significativamente la dimensionalidad sin recurrir a técnicas como ACP.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.5750	0.5750	0.5641
Mediana	0.5752	0.5752	0.5654
Desviación Estándar	0.0047	0.0047	0.0303
Min	0.5664	0.5664	0.4769
Max	0.5851	0.5851	0.6061

Tabla 12: Resultados estadísticos de NEAT utilizando *Bag of Words* reducida (335 palabras)

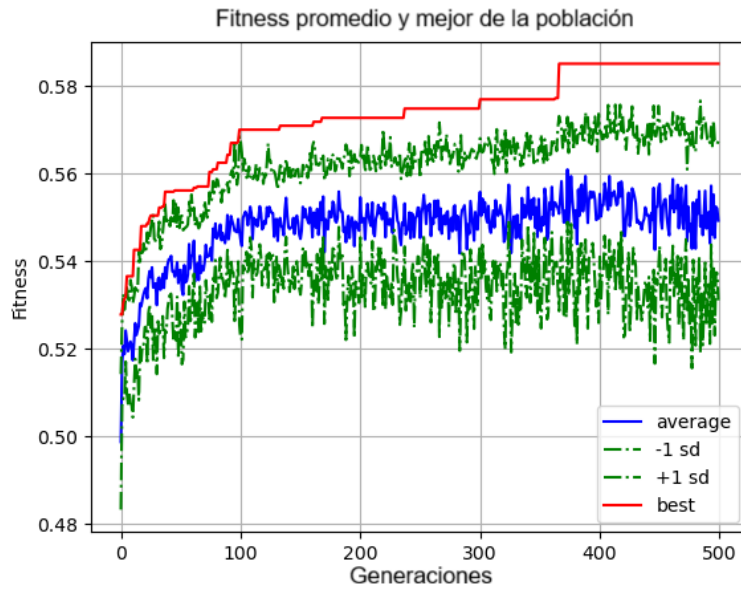


Figura 14: Gráfico de convergencia del Mejor Fitness con *Bag of Words* Reducida (335)

b. Original

La Tabla 13 presenta los resultados obtenidos al aplicar NEAT con la representación *Bag of Words* original (846 características). A diferencia de las tablas anteriores, esta muestra los resultados individuales de tres ejecuciones experimentales independientes, variando tanto la semilla aleatoria como el número de generaciones, esto debido al alto costo computacional que implicaban las ejecuciones.

Representación	Nº de características	Semilla	Nº de generaciones	Fitness	Train accuracy	Validation accuracy
BoW Binaria	846	15	200	0.6022	0.6022	0.5276
		14	500	0.6110	0.6110	0.5726
		13	500	0.6083	0.6083	0.5465

Tabla 13: Resultados Estadísticos de NEAT con *Bag of Words* Original

Se puede observar que, a medida que aumenta el número de generaciones, el desempeño mejora ligeramente, alcanzando un *fitness* y *train accuracy* máximos de 0.6110, y un *validation accuracy* de 0.5726 con la semilla 14 y 500 generaciones. Estos resultados indican que, aunque esta representación es más costosa computacionalmente que la reducida, permite capturar mayor información del corpus, lo que se refleja en un mejor rendimiento en validación.

A lo largo de las ejecuciones, la mejor exactitud de validación se obtuvo en el experimento con 500 generaciones. El gráfico de convergencia (ver Figura 15) indica que el mejor *fitness* se estabiliza cerca de 0.61 después de las primeras 150 generaciones, mostrando fluctuaciones moderadas en el *fitness* promedio a lo largo de las 500 generaciones. Estas fluctuaciones reflejan la complejidad del proceso evolutivo al trabajar con una alta dimensionalidad, lo que sugiere que el modelo puede beneficiarse de técnicas de reducción de dimensionalidad para lograr una convergencia más estable y eficiente.

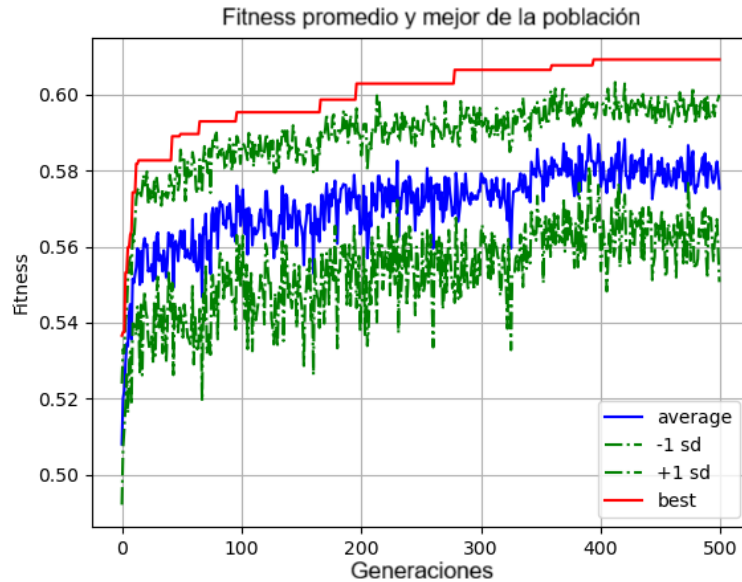


Figura 15: Gráfico de convergencia del Mejor Fitness con *Bag of Words* Original

6.3.3. Resultados de NEAT con TF-IDF

La representación TF-IDF, que consta de 846 características, tiene un rendimiento inferior en comparación con las anteriores representaciones de texto. Como

se muestra en la Tabla 14, el *validation accuracy* promedio alcanzado fue de 0.5298, lo que refleja una mayor variabilidad en los resultados obtenidos. Los valores extremos de *validation accuracy* oscilaron entre 0.4942 y 0.5262, evidenciando un menor rango de desempeño en comparación con las configuraciones basadas en BoW.

El mejor *fitness* se estabilizó alrededor de 0.6 después de las primeras 150 generaciones, con fluctuaciones moderadas en el *fitness* promedio (ver Figura 16). Sin embargo, estas fluctuaciones fueron más pronunciadas en comparación con representaciones anteriores.

Métrica	Fitness	Train Accuracy	Validation Accuracy
Promedio	0.6016	0.6016	0.5298
Mediana	0.6017	0.6017	0.5262
Desviación Estándar	0.0055	0.0055	0.0239
Min	0.5945	0.5945	0.4962
Max	0.6128	0.6128	0.5262

Tabla 14: Resultados estadísticos utilizando la representación TF-IDF

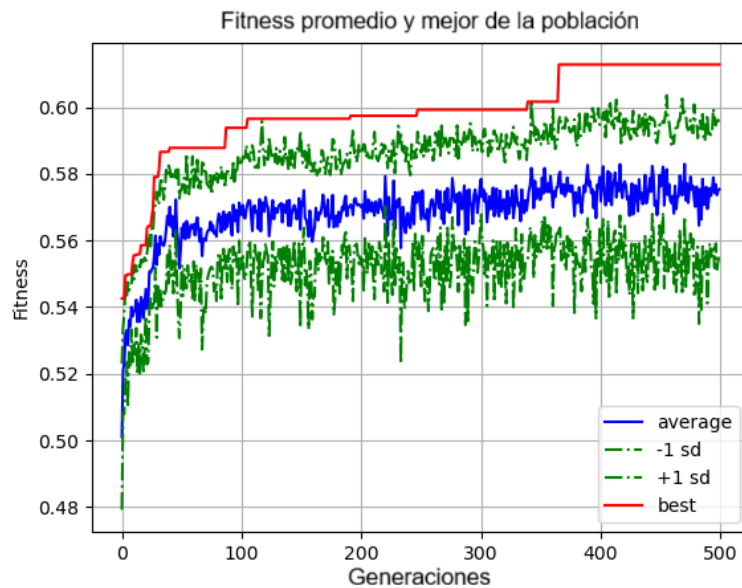


Figura 16: Gráfico de convergencia del Mejor Fitness con TF-IDF

6.3.4. Comparación de desempeño de NEAT con diferentes representaciones de texto

En la Tabla 15 se presenta una comparativa global del desempeño de NEAT utilizando las diferentes representaciones de texto, incluyendo las configuraciones evaluadas con ACP, BoW reducida y BoW limpia, así como las basadas en embeddings de BERT y TF-IDF. Es importante señalar que no se incluyen los resultados obtenidos con la representación *Bag of Words* original, ya que no era directamente comparable con las demás configuraciones al haberse realizado únicamente tres

ejecuciones experimentales. Estos resultados se presentan y analizan en su sección correspondiente (véase Sección b.). Por lo tanto, esta tabla concentra únicamente los resultados obtenidos tras realizar 15 ejecuciones por configuración para cada representación de texto, lo que permite una comparación más robusta y consistente entre las distintas estrategias de representación.

Tabla 15: Desempeño de NEAT con diferentes representaciones de texto. Entre paréntesis se indica el número de características.

Representación de texto	Fitness	Train Accuracy	Validation Accuracy
ACP (50)	0.6064	0.6064	0.6024
ACP (100)	0.6092	0.6092	0.5989
ACP (200)	0.6122	0.6122	0.5809
ACP (278)	0.6104	0.6104	0.5653
BoW reducida (335)	0.5750	0.5750	0.5641
BoW limpia (2860) + ACP (50)	0.6084	0.6084	0.6150
BERT (768)	0.5859	0.5859	0.5209
BERT sin preprocesamiento (768)	0.6450	0.6450	0.6244
TF-IDF (846)	0.6016	0.6016	0.5298

Los resultados obtenidos muestran que el mejor *validation accuracy* para el algoritmo NEAT se alcanzó utilizando la representación BERT sin preprocesamiento, logrando un valor de 0.6244. Esta representación también produjo el mayor valor de *fitness* y *train accuracy* (0.6450), lo que refleja la riqueza contextual de los embeddings generados directamente desde el texto original, sin necesidad de limpieza previa. Por otro lado, la representación BoW limpia reducida con ACP a 50 componentes también demostró un rendimiento competitivo (0.6150), superando incluso a todas las configuraciones tradicionales de ACP basadas en la BoW original. Esto confirma que un preprocesamiento cuidadoso seguido de reducción dimensional puede mejorar la capacidad de generalización del modelo. En contraste, las representaciones con ACP de mayor dimensión (200 y 278 componentes) no lograron superar los resultados obtenidos con representaciones más compactas, lo cual sugiere que no siempre una mayor cantidad de componentes garantiza un mejor desempeño. Por otra parte, el tiempo de entrenamiento varió de forma considerable según la representación utilizada: los experimentos con BoW presentaron los mayores tiempos de cómputo, alcanzando ejecuciones de días en algunos casos, mientras que las configuraciones con reducción de dimensionalidad mediante ACP redujeron significativamente la duración, con ejecuciones en el orden de varias horas. Esta diferencia se debe principalmente a la complejidad computacional asociada al tamaño del espacio de características en cada representación. *Nota: Estos tiempos corresponden a estimaciones aproximadas, obtenidas a partir de la observación empírica durante la ejecución de los experimentos, ya que no se registraron mediciones exactas de duración.*

Para validar los resultados se realizaron pruebas estadísticas que permitieran determinar si las diferencias observadas entre las representaciones de texto eran estadísticamente significativas o si podían deberse al azar. En primer lugar, se aplicó la prueba de Shapiro–Wilk con el objetivo de verificar si las distribuciones de *Vali-*

dation Accuracy cumplieran con los supuestos paramétricos requeridos para este tipo de análisis. Los resultados mostraron que todas las distribuciones presentaron p-valores mayores a 0.05 (ver Tabla 16). Una vez confirmado este supuesto, se aplicó un ANOVA de un factor sobre dicha métrica, ya que esta prueba resulta adecuada para comparar más de dos grupos de manera simultánea, y contrasta la hipótesis nula de que todas las representaciones presentan el mismo desempeño medio contra la alternativa de que al menos una difiere de manera significativa.

Tabla 16: Resultados de la prueba de normalidad Shapiro–Wilk.

Representación	p-valor
PCA (50)	0.0821
PCA (100)	0.2577
PCA (200)	0.8994
PCA (278)	0.3026
BoW reducida (335)	0.0684
BoW limpia (2860) + PCA (50)	0.3701
BERT (768)	0.4313
BERT sin preprocesamiento	0.3733
TF-IDF (846)	0.4214

Tabla 17: Resultados de las pruebas estadísticas sobre *Validation Accuracy*. El ANOVA indicó diferencias significativas entre representaciones ($p < 0.001$). Los grupos se obtuvieron con la prueba post-hoc de Tukey HSD.

Representación de texto	Grupo Tukey
ACP (50)	A–B
ACP (100)	B
ACP (200)	B–C
ACP (278)	C
BoW reducida (335)	C
BoW limpia (2860) + ACP (50)	A
BERT (768)	D
BERT sin preprocesamiento (768)	A
TF-IDF (846)	D

El resultado del ANOVA fue estadísticamente significativo ($F(8, 126) = 42.07$, $p = 4.71 \times 10^{-32}$), lo cual indica que el tipo de representación de texto influye en el desempeño del modelo NEAT y que existen diferencias reales entre los grupos. Sin embargo, el ANOVA por sí mismo no señala entre qué representaciones se encuentran dichas diferencias, únicamente confirma su existencia global. Por esta razón, se aplicó una prueba post-hoc de **Tukey HSD (Honestly Significant Difference)**. Esta prueba permite realizar comparaciones múltiples por pares controlando el error de tipo I, de manera que se puede identificar con confianza qué representaciones difieren

significativamente entre sí.

Los resultados de esta prueba, resumidos en la Tabla 17, mostraron que BERT sin preprocesamiento (768) y BoW limpia (2860) + PCA (50) forman el grupo con mejor desempeño, sin diferencias significativas entre ellas y destacándose por encima del resto. En contraste, BERT (768) y TF-IDF (846) se ubicaron en el grupo con menor desempeño, sin diferencia significativa entre ambos, pero con resultados claramente inferiores frente al resto de representaciones. Las configuraciones basadas en PCA (50, 100, 200, 278) y BoW reducida (335) ocuparon posiciones intermedias, mostrando diferencias entre sí según el número de componentes o características conservadas.

6.3.5. Resultados de NEAT con SVM

Para evaluar el desempeño de SVM, se aplicaron las mismas representaciones de texto utilizadas con NEAT, incluyendo las variantes reducidas mediante ACP, las versiones con limpieza profunda del corpus y las representaciones generadas con BERT. Los resultados se presentan en la Tabla 18, que muestra los valores de accuracy obtenidos por SVM en los conjuntos de entrenamiento y validación para cada representación.

Tabla 18: Desempeño de SVM con diferentes representaciones de texto. Entre paréntesis se indica el número de características en cada caso.

Representación de texto	Train Accuracy	Validation Accuracy
ACP (50)	0.6023	0.6337
ACP (100)	0.6345	0.6206
ACP (200)	0.6538	0.6032
ACP (278)	0.6812	0.5799
BoW reducida (335)	0.6270	0.4811
BoW limpia (2860) + ACP (50)	0.5977	0.6299
BERT (768)	0.7469	0.5596
BERT sin preprocesamiento (768)	0.8030	0.6633
TF-IDF (846)	0.9238	0.6122

Se puede observar que la representación basada en BERT sin preprocesamiento logró el mayor accuracy en validación (0.6633), superando a todas las demás representaciones utilizadas. Este resultado indica que SVM, puede beneficiarse considerablemente de la riqueza contextual de los embeddings densos generados por BERT, especialmente cuando se preserva el texto en su forma original. Esta representación parece capturar información semántica suficiente para permitir que un clasificador lineal logre un rendimiento competitivo en una tarea compleja como la detección de ironía. Asimismo, la versión reducida mediante ACP de *Bag of Words* con limpieza profunda (compuesta originalmente por 2,860 términos) también obtuvo un desempeño notable, alcanzando un accuracy de validación de 0.6299. Este resultado respalda la efectividad del preprocesamiento adicional y la reducción de dimensionalidad como estrategias que mejoran la capacidad de generalización de modelos clásicos como SVM. Esta representación superó a todas las demás configuraciones de ACP, incluso aquellas con mayor número de componentes (100, 200 y 278), lo que sugiere que la

calidad de las características seleccionadas puede ser más determinante que su cantidad. Los resultados permiten observar cómo un modelo como SVM puede adaptarse a diversas representaciones y aún así mantener un rendimiento competitivo. No obstante, estas representaciones deben ser cuidadosamente seleccionadas, pues un exceso de ruido como una representación subóptima pueden deteriorar significativamente el rendimiento.

7. Análisis de las Redes Evolucionadas con NEAT

En esta sección se presenta un análisis estructural y de desempeño de las redes neuronales evolucionadas con NEAT, con el fin de comprender cómo varían sus características internas, su complejidad y su rendimiento según las diferentes configuraciones evaluadas al final del proceso evolutivo.

7.1. Conexiones y Estructura de Redes Evolucionadas en NEAT

Uno de los aspectos más distintivos del algoritmo NEAT es su capacidad para evolucionar simultáneamente tanto la topología como los pesos de las redes neuronales, lo que permite generar arquitecturas adaptativas y eficientes sin requerir una estructura fija predefinida. En la Tabla 19 se presentan las estadísticas de los pesos de las conexiones habilitadas y deshabilitadas hacia la capa de salida para diferentes configuraciones de componentes. Estos valores permiten analizar la distribución y variabilidad de las conexiones según la representación utilizada.

Tabla 19: Estadísticas de Conexiones Habilitadas y Deshabilitadas hacia la Capa de Salida

Nodo de salida	Estado de Conexión	Promedio	Varianza	Mínimo	Máximo
50 Componentes					
0	Habilitadas	-0.2930	3.8281	-5.0	5.0
0	Deshabilitadas	0.0390	2.9527	-5.0	4.2
1	Habilitadas	0.1729	3.2819	-5.0	5.0
1	Deshabilitadas	0.0702	2.3433	-4.1	3.6
100 Componentes					
0	Habilitadas	-0.2841	3.5073	-5.0	5.0
0	Deshabilitadas	-0.0965	2.4317	-5.0	5.0
1	Habilitadas	0.0279	3.0194	-5.0	5.0
1	Deshabilitadas	-0.0248	2.5176	-4.3	4.9
200 Componentes					
0	Habilitadas	-0.0427	3.0807	-5.0	5.0
0	Deshabilitadas	-0.0137	2.6126	-5.0	5.0
1	Habilitadas	-0.0020	3.0061	-5.0	5.0
1	Deshabilitadas	0.0171	2.3085	-5.0	5.0

Las **Conexiones habilitadas** presentan, en general, un rango amplio de valores de peso, lo que sugiere que durante el proceso de evolución, NEAT prioriza estas conexiones al asignarles un impacto significativo en la predicción final. En la configuración con 50 componentes, los pesos de las conexiones habilitadas hacia el nodo de salida 1 tienen un promedio de 0.1729 y una varianza de 3.2819, lo que indica una alta diversidad en la magnitud de los pesos. Por otro lado, las **Conexiones deshabilitadas** muestran menores rangos y varianza. Esto refuerza la hipótesis de que NEAT tiende a desactivar aquellas conexiones cuyo impacto es no significativo o contraproducente en el modelo final.

En las Figuras 17, 18 y 19, se muestra la distribución de conexiones habilitadas y deshabilitadas hacia la capa de salida en los experimentos con 50, 100 y 200 componentes, mostrando cómo varía el número de conexiones en función de la semilla utilizada.

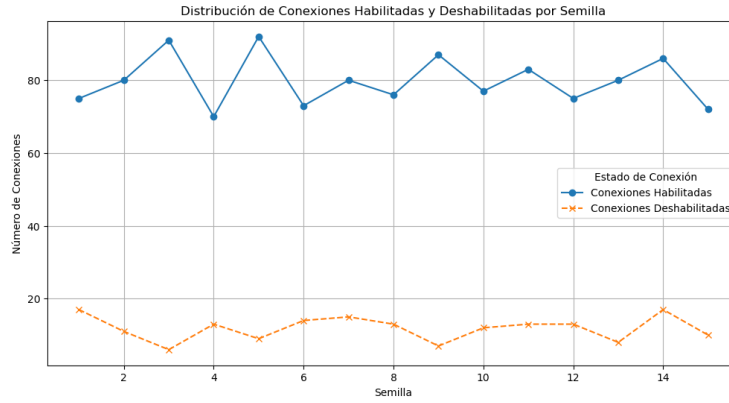


Figura 17: 50 Componentes

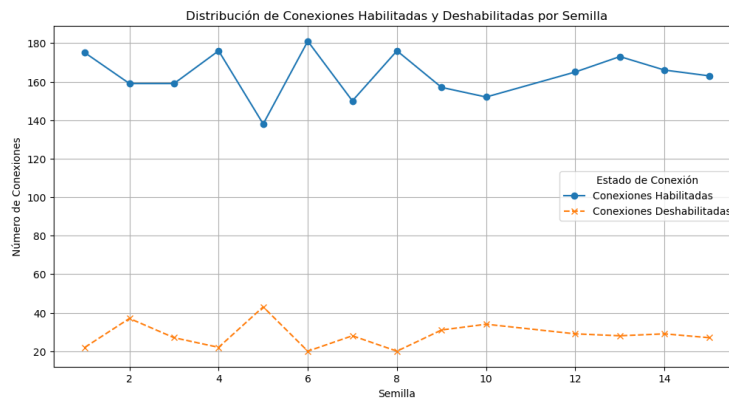


Figura 18: 100 Componentes

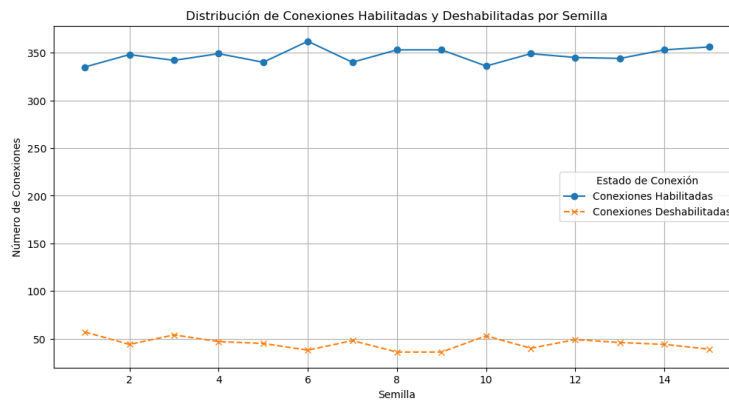


Figura 19: 200 Componentes

En el gráfico correspondiente a **50 componentes**, se observa que el número de conexiones habilitadas varía entre 60 y 90, mientras que las deshabilitadas se mantienen consistentemente en un rango de 10 a 20 conexiones. La variabilidad en el número de conexiones habilitadas sugiere que, con un menor número de componentes, NEAT tiende a generar diferentes configuraciones topológicas dependiendo de la semilla, lo que implica una mayor sensibilidad a la aleatoriedad inicial. En el caso

de **100 componentes**, el número de conexiones habilitadas oscila entre 160 y 180, mientras que las deshabilitadas se mantienen en un rango de 20 a 30 conexiones. Aquí se aprecia una menor variabilidad en comparación con la configuración de 50 componentes, lo que sugiere que aumentar el número de componentes contribuye a estabilizar el proceso evolutivo, generando redes neuronales más consistentes entre diferentes semillas. En la configuración de **200 componentes**, el número de conexiones habilitadas se encuentra entre 340 y 360, mientras que las deshabilitadas varían ligeramente entre 40 y 50 conexiones.

A medida que se incrementa el número de componentes principales, la variabilidad en el número de conexiones habilitadas y deshabilitadas disminuye, lo que indica una mayor consistencia en las redes neuronales generadas por NEAT. Esto se traduce en una menor sensibilidad del modelo a la semilla inicial, lo que es deseable en términos de estabilidad y reproducibilidad de los resultados.

7.2. Distancia Genómica y Desempeño del Modelo

Esta sección es orientada a analizar la estructura topológica de las redes neuronales evolucionadas mediante NEAT y su desempeño en tareas de clasificación. El análisis se fundamenta en los resultados experimentales obtenidos a partir de una representación del texto mediante BoW Binaria, la cual fue sometida a una limpieza exhaustiva y posteriormente reducida dimensionalmente mediante ACP a 50 componentes. A partir de esta configuración, se realizaron un total de 15 ejecuciones independientes del algoritmo NEAT, cada una con una población de 20 genomas. El objetivo central de este análisis es explorar cómo varía el desempeño de las redes en función de su proximidad estructural con respecto al genoma de mejor rendimiento en cada ejecución, medida a través de la distancia genómica, una función integrada en NEAT que cuantifica qué tan diferentes son dos genomas en términos de su topología y pesos. Esta medida considera los genes disjuntos y excedentes, así como las diferencias de pesos entre genes coincidentes. A partir de esta métrica, en cada ejecución se identificó el genoma con la menor distancia genómica respecto al ganador, permitiendo evaluar el rendimiento de redes estructuralmente similares al mejor modelo. Al finalizar cada ejecución, se extrajo el genoma ganador (aquel con el mayor valor de *fitness* en los datos de entrenamiento) y se comparó su estructura con las otras 19 redes neuronales presentes en la última generación.

La distancia genómica d surge a partir de las mutaciones que introducen variación estructural en los genomas. En NEAT, estas mutaciones consisten principalmente en añadir nuevas conexiones o nuevos nodos, lo cual genera genes con identificadores de innovación inéditos. Dichos genes, al ser comparados con otros genomas, se clasifican como *disjoint* o *excess*, dependiendo de si se encuentran dentro o fuera del rango de genes coincidentes. La Figura 20 ejemplifica este proceso: en la parte superior se observa la mutación por adición de conexión, mientras que en la parte inferior se muestra la mutación por adición de nodo. Ambos casos generan nuevos genes que, al no estar presentes en otros genomas, incrementan los valores de D (*disjoint*) o E (*excess*) en la fórmula de la distancia genética (Ver Ecuación 12). Por su parte, los genes que coinciden entre genomas permiten calcular la diferencia promedio de pesos $\overline{W}$, completando así los tres componentes de la métrica.

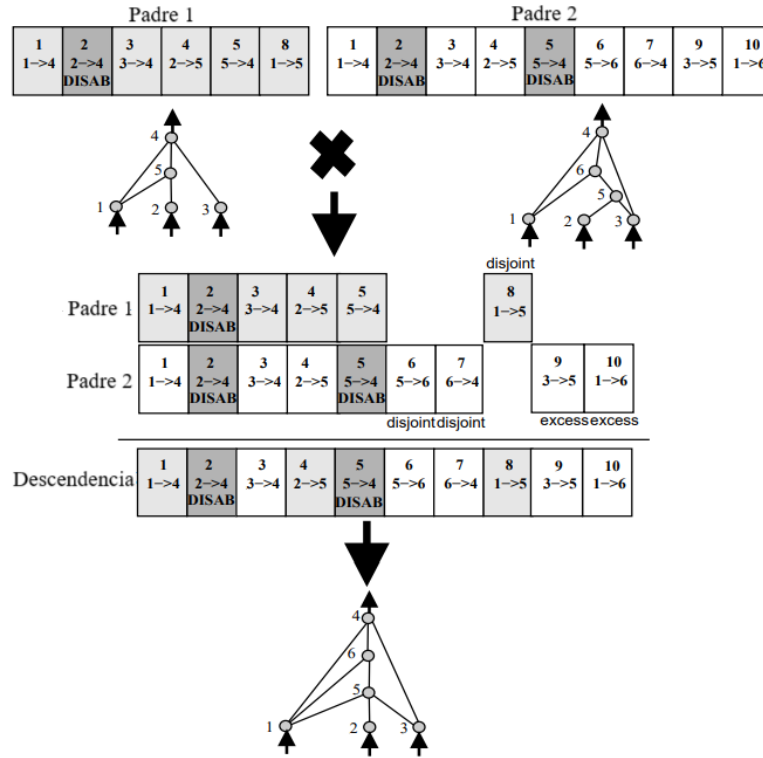


Figura 20: Ejemplo de mutaciones en NEAT: adición de conexión (arriba) y adición de nodo (abajo). Estas mutaciones generan nuevos genes que, al compararse entre genomas, se clasifican como *disjoint* o *excess*, contribuyendo directamente al cálculo de la distancia genómica. Adaptado de [39].

En cada una de las 15 ejecuciones independientes se seleccionó el genoma más cercano al ganador con base en la menor distancia genómica. Posteriormente, se analizaron sus respectivos resultados de validación, así como la cantidad de conexiones habilitadas, con el fin de estudiar la relación entre complejidad estructural, similitud genómica y desempeño predictivo.

Las Figuras 21 y 22 muestran los resultados obtenidos para la ejecución número 10, la cual obtuvo el mayor valor de accuracy en validación entre todas las ejecuciones. En ambas gráficas, el punto rojo destaca el genoma con menor distancia genómica al ganador.

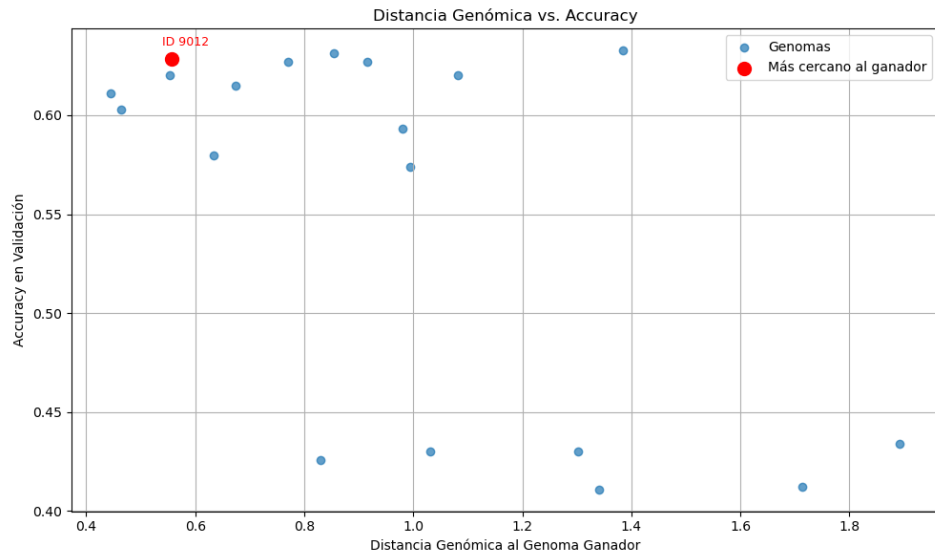


Figura 21: Relación entre la distancia genómica y el accuracy de validación. El punto rojo representa el genoma más cercano al ganador.

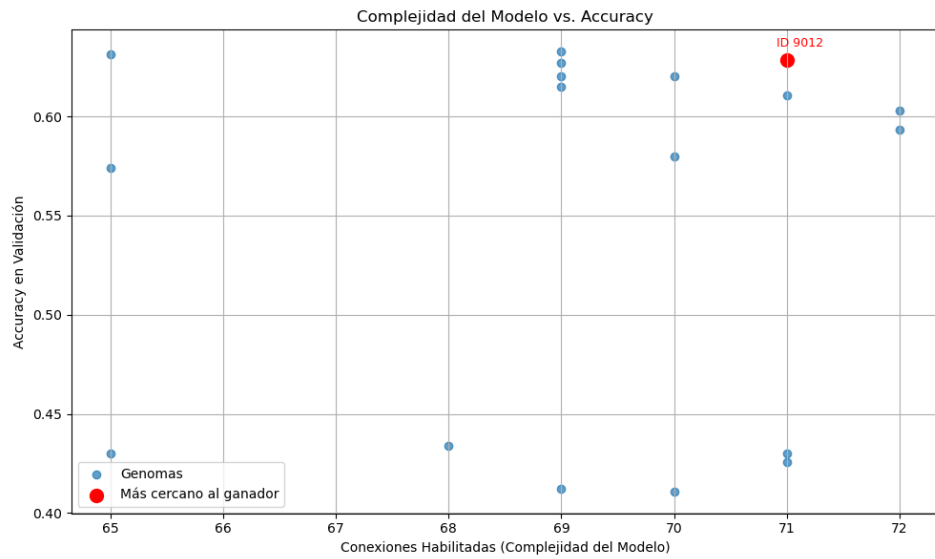


Figura 22: Relación entre la cantidad de conexiones habilitadas y el accuracy en validación. El punto rojo representa el genoma más cercano al ganador.

Como se observa en ambas figuras, el genoma más cercano estructuralmente al ganador (ID 9012) se encuentra dentro del grupo de redes con mejor desempeño, lo que sugiere que la proximidad genómica puede reflejarse en comportamientos similares a nivel de predicción. Además, este genoma presenta una complejidad comparable (medida en conexiones habilitadas), lo que refuerza la idea de que NEAT tiende a concentrar soluciones óptimas en regiones específicas del espacio de búsqueda.

Sin embargo, dado que el coeficiente de Pearson requiere que las variables analizadas sigan una distribución normal, se aplicaron pruebas de normalidad utilizando

el test de Shapiro–Wilk. Los resultados indicaron que ninguna de las variables cumple con el supuesto de normalidad ($p < 0.05$ en todos los casos), lo cual es consistente con la naturaleza acotada, discreta y frecuentemente asimétrica de los datos generados por algoritmos evolutivos como NEAT. Debido a ello, la interpretación principal del análisis correlacional se basa en los coeficientes de Spearman, una alternativa no paramétrica adecuada para evaluar relaciones monótonas sin requerir normalidad. En la Tabla 20 se reportan los valores de p obtenidos en las pruebas de normalidad, mientras que la Tabla 21 resume los coeficientes de correlación resultantes a lo largo de las 15 ejecuciones.

Para evaluar de manera más robusta la relación entre distancia genómica, complejidad del modelo y accuracy, se calcularon coeficientes de correlación de Pearson y Spearman para cada ejecución. Sin embargo, dado que el coeficiente de Pearson requiere que las variables analizadas sigan una distribución normal, se aplicaron pruebas de normalidad utilizando el test de Shapiro–Wilk. Los resultados indicaron que ninguna de las variables cumple con el supuesto de normalidad ($p < 0.05$ en todos los casos). Debido a ello, la interpretación principal del análisis correlacional se basa en los coeficientes de Spearman, una alternativa no paramétrica para evaluar relaciones monótonas sin requerir normalidad. En la Tabla 20 resume los valores promedio de estos coeficientes a lo largo de las 15 ejecuciones.

Tabla 20: Correlación entre características estructurales y validation accuracy a lo largo de 15 ejecuciones con NEAT.

Tipo de correlación	Promedio	Desviación estándar	Correlaciones significativas ($p < 0.05$)
Spearman: Distancia Genómica vs. Accuracy	-0.0917	0.2190	0 de 15
Spearman: Complejidad del Modelo vs. Accuracy	0.0928	0.1819	0 de 15

Los resultados estadísticos indican que no existe una correlación fuerte ni estadísticamente significativa entre las características estructurales de las redes evolucionadas (medidas mediante la distancia genómica y el número de conexiones habilitadas) y el accuracy obtenido en validación. Por otro lado, al analizar la complejidad del modelo (representada por el número de conexiones habilitadas), las correlaciones con el accuracy en validación también fueron bajas, lo que sugieren que tener una red con mayor cantidad de conexiones no implica necesariamente un mejor desempeño en la tarea de clasificación. En otras palabras, una mayor complejidad estructural no garantiza una mayor capacidad predictiva en este contexto. En conjunto, el rendimiento de las redes neuronales evolucionadas mediante NEAT no parece depender directamente ni de su cercanía estructural al mejor modelo ni de su número total de conexiones habilitadas.

8. Conclusiones y Trabajo Futuro

Este trabajo exploró la aplicación del algoritmo NEAT en la tarea de detección de ironía en *tweets*, comparando su rendimiento con un clasificador tradicional SVM utilizando distintas representaciones del texto: *Bag of Words* Binaria, TF-IDF, vectores reducidos mediante ACP y representaciones densas generadas por BERT.

Los resultados obtenidos evidencian que NEAT es capaz de adaptarse a espacios de características complejos siempre que la representación del texto sea adecuadamente seleccionada y procesada. En particular, la reducción dimensional aplicada a una representación BoW profundamente depurada permitió mejoras sustanciales en el rendimiento, lo que resalta la importancia de la calidad del corpus y del preprocesamiento en la eficacia de modelos evolutivos. Además, el uso de embeddings de BERT sin ningún tipo de preprocesamiento textual condujo al mayor *validation accuracy* obtenido por NEAT, lo que sugiere que el algoritmo puede beneficiarse de espacios semánticamente ricos cuando se preserva el contexto lingüístico original.

Aunque NEAT no superó consistentemente al clasificador SVM en todas las configuraciones, logró un rendimiento comparable e incluso superior en ciertos escenarios, particularmente cuando el espacio de entrada fue preprocesado cuidadosamente o cuando se emplearon representaciones densas como BERT sin preprocesamiento. Estos hallazgos no posicionan a NEAT como un sustituto directo de los clasificadores tradicionales, sino que subrayan su potencial como enfoque evolutivo adaptable, especialmente útil cuando la estructura óptima de la red no se conoce a priori.

Uno de los aportes clave de esta investigación fue la inclusión de un análisis estructural basado en la distancia genómica entre redes neuronales generadas en la población final. La evidencia obtenida refuerza la noción de que los enfoques evolutivos pueden ofrecer ventajas cualitativas más allá del rendimiento numérico, proporcionando información estructural que los modelos tradicionales no revelan. Esta capacidad de inspección del proceso de adaptación en sí mismo convierte a NEAT en una herramienta de investigación valiosa para comprender cómo los modelos procesan fenómenos lingüísticos complejos, especialmente aquellos que dependen de señales pragmáticas o discursivas difíciles de capturar mediante métodos deterministas.

A pesar de los beneficios observados, el uso de NEAT implica un mayor costo computacional en comparación con modelos tradicionales, debido al proceso evolutivo que involucra múltiples generaciones y la evaluación de diversas arquitecturas. Este aspecto plantea un reto importante, especialmente en contextos donde se requiere eficiencia en tiempo de entrenamiento. Para el trabajo futuro, se propone profundizar en varias líneas de investigación: (1) Incorporar representaciones más sofisticadas del texto, como variantes ajustadas de BERT o modelos multilingües, que podrían capturar mejor los matices de la ironía; (2) Implementar técnicas de preprocesamiento avanzadas o adaptativas que puedan ajustarse a las particularidades del corpus; (3) Optimizar el proceso evolutivo de NEAT mediante estrategias híbridas que combinen aprendizaje supervisado con evolución estructural para reducir el costo computacional; y (4) Analizar el impacto de características semánticas individuales en las conexiones y pesos finales de las redes evolucionadas.

Otra línea de investigación relevante consiste en extender el análisis estructural incorporando medidas adicionales —como agrupamientos funcionales dentro de la red evolucionada— que permitan caracterizar con mayor precisión cómo emergen patrones especializados en la detección de ironía. Esto podría revelar si ciertas configuraciones neuronales recurrentes están asociadas a comportamientos predictivos superiores, aportando evidencia sobre la convergencia evolutiva hacia subestructuras óptimas. Además, la integración de NEAT con técnicas modernas de representación, como embeddings contextuales ajustados mediante *fine-tuning*, abre la posibilidad de explorar sistemas neuroevolutivos híbridos que combinen aprendizaje profundo con exploración evolutiva. Dichos sistemas podrían evaluar arquitecturas de mayor profundidad o topologías con mecanismos de atención, lo que permitiría estudiar si la evolución es capaz de descubrir estructuras funcionales comparables a las empleadas manualmente en modelos transformer.

Finalmente, se destaca que NEAT constituye una herramienta poderosa no solo por su capacidad de clasificación, sino también como marco experimental para estudiar cómo la representación del texto, la dimensionalidad y el preprocesamiento afectan la arquitectura y comportamiento de modelos adaptativos en tareas complejas de PLN como la detección de ironía. Se espera que, con los ajustes y optimizaciones propuestos, un algoritmo como NEAT pueda demostrar obtener resultados similares o mejores que los logrados por las propuestas del Estado del Arte. Esto permitiría consolidar su uso como una alternativa competitiva en tareas de clasificación de ironía y problemas complejos de Procesamiento de Lenguaje Natural.

Referencias

- [1] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459.
- [2] Attardo, S. (2000). *Irony as Relevant Inappropriateness*. John Benjamins Publishing.
- [3] Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).
- [4] Bamman, D. and Smith, N. A. (2015). Contextualized sarcasm detection on twitter. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media (ICWSM 2015)*, pages 574–577.
- [5] Barbieri, F., Kruszewski, G., Ronzano, F., and Saggion, H. (2016). How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 10th International Conference on Web and Social Media (ICWSM)*.
- [6] Bishop, C. M. (2003). *Pattern Recognition and Machine Learning*. Springer.
- [7] Bosco, C., Patti, V., and Pinasco, B. (2013). Developing corpora for sentiment analysis and opinion mining: The case of irony and sarcasm. *Language Resources and Evaluation*, 47(1):283–308.
- [8] Burgers, C. F. (2010). *Verbal Irony: Use and Effects in Written Discourse*. PhD thesis, Vrije Universiteit Amsterdam / Radboud University Nijmegen. Dissertation.
- [9] Cai, Y., Cai, H., and Wan, X. (2019). Multimodal sarcasm detection in twitter with hierarchical fusion model. In *Proceedings of ACL*, pages 2506–2515.
- [10] Carvalho, P., Sarmiento, L., Silva, M. J., and de Oliveira, E. (2009). Clues for detecting irony in user-generated contents: oh...!! it's "so easy";-). In *Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, TSA '09, page 53–56, New York, NY, USA. Association for Computing Machinery.
- [11] Clark, H. H. and Gerrig, R. (1984). On the pretense theory of irony. *Journal of experimental psychology. General*, 113 1:121–6.
- [12] Davidov, D., Tsur, O., and Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL)*, pages 107–116.
- [13] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- [14] Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer.

- [15] Farías, D. I. H. (2017). *Irony and Sarcasm Detection in Twitter: The Role of Affective Content*. PhD thesis, Universitat Politècnica de València.
- [16] Farías, D. I. H., Patti, V., and Rosso, P. (2016). Irony detection in twitter: The role of affective content. 16(3).
- [17] Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. MIT Press, Cambridge, MA.
- [18] Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. In *CS224N Project Report, Stanford*.
- [19] González-Ibáñez, R., Muresan, S., and Wacholder, N. (2011). Identifying sarcasm in twitter: A closer look. pages 581–586.
- [20] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422.
- [21] Hernandez-Farias, D. I., Prati, R. C., Herrera, F., and Rosso, P. (2020). Irony detection in twitter with imbalanced class distributions. *Journal of Intelligent Fuzzy Systems*, 39(2):2147–2163.
- [22] Hernández-Hernández, J.-C., Mezura-Montes, E., Hoyos-Rivera, G.-d.-J., and Rodríguez-López, O. (2021). Neuroevolution for sentiment analysis in tweets written in mexican spanish. In *Pattern Recognition*, pages 101–110. Springer International Publishing.
- [23] Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.
- [24] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142. Springer.
- [25] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065):1–16.
- [26] Kadamala, K. and Griffith, J. (2023). An insight into neuroevolution and genetic algorithms for text classification. *Procedia Computer Science*, 225:1379–1387. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Systems (KES 2023).
- [27] Kwok, I. and Wang, Y. (2013). Locate the hate: Detecting tweets against blacks. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.
- [28] Martín del Brío, B. and Sanz Molina, A. (2001). *Redes neuronales y sistemas difusos*. Alfaomega, México, 2 edition.
- [29] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

- [30] Mosbach, M., Andriushchenko, M., and Klakow, D. (2021). On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.
- [31] Ortega Bueno, R., Rangel Pardo, F. M., Hernández Farías, D. I., Rosso, P., Montes-y Gómez, M., and Medina-Pagola, J. E. (2019). Overview of the task on irony detection in spanish variants. In *Proceedings of IberLEF 2019 at SEPLN*.
- [32] Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 1320–1326.
- [33] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.
- [34] Potamias, R. A., Siolas, G., and Stafylopatis, A. (2020). A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32:17309–17320.
- [35] Ramos, J. E. (2003). Using tf-idf to determine word relevance in document queries.
- [36] Reyes Antonio, R. P. and Tony, V. (2016). A multidimensional approach for detecting irony in twitter. *Language Resources and Evaluation*.
- [37] Riloff, E., Qadir, A., Surve, P., Silva, L. D., Gilbert, N., and Huang, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation. In *Conference on Empirical Methods in Natural Language Processing*.
- [38] Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*.
- [39] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- [40] Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*.
- [41] Tepperman, J., Traum, D., and Narayanan, S. (2006). Yeah right: Sarcasm recognition for spoken dialogue systems. In *Proc. INTERSPEECH*, pages 1838–1841.
- [42] Van Hee, C., Lefever, E., and Hoste, V. (2018). Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 39–50.
- [43] Wallace, B. C., Choe, D. K., Kertz, L., and Charniak, E. (2014). Humans require context to infer ironic intent (so computers probably do, too). In Toutanova, K. and Wu, H., editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516, Baltimore, Maryland. Association for Computational Linguistics.

- [44] Zhang, Z., David, R., and Jonathan, T. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. *Springer International Publishing*.