Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado
Facultad de Ingeniería Industrial
Unidad de Posgrado

Evaluación del uso de redes sociales y su impacto sobre la elección presidencial peruano 2021 mediante técnicas de Machine Learning

TESIS

Para optar el Grado Académico de Doctor en Ingeniería Industrial

AUTOR

Eber Joseph BALLON ALVAREZ

ASESOR

Dr. Juan Manuel CEVALLOS AMPUERO

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Ballon, E. (2026). *Evaluación del uso de redes sociales y su impacto sobre la elección presidencial peruano 2021 mediante técnicas de Machine Learning*. [Tesis de doctorado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Industrial, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Eber Joseph Ballon Alvarez
Tipo de documento de identidad	DNI
Número de documento de identidad	02441461
URL de ORCID	https://orcid.org/**0000-0003-2362-5743**
Datos de asesor	
Nombres y apellidos	Juan Manuel Cevallos Ampuero
Tipo de documento de identidad	DNI
Número de documento de identidad	07855059
URL de ORCID	https://orcid.org/0000-0001-8612-9128
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Francisco Javier Wong Cabanillas
Tipo de documento	DNI
Número de documento de identidad	07564428
Miembro del jurado 1	
Nombres y apellidos	Pedro Pablo Rosales López
Tipo de documento	DNI
Número de documento de identidad	10419269
Miembro del jurado 2	
Nombres y apellidos	Eduardo Walter Buse Thorne
Tipo de documento	DNI
Número de documento de identidad	07820784

Miembro del jurado 3	
Nombres y apellidos	José Ovidio Flores Gutiérrez
Tipo de documento	CE
Número de documento de identidad	VE / 2318980
Datos de investigación	
Línea de investigación	1. Inteligencia artificial. ODS 9: Industria, innovación e infraestructura. C.17.0.1
Grupo de investigación	NO APLICA
Agencia de financiamiento	SIN FINANCIAMIENTO
Ubicación geográfica de la investigación	País: Perú Departamento de Lima Provincia: Lima Distrito: San Borja Lugar: Av. San Borja Norte 323 Latitud: -12.0942261 Longitud: -77.0071006
Año o rango de años en que se realizó la investigación	2021 -2026
URL de disciplinas OCDE	Ingeniería, Tecnología https://purl.org/pe-repo/ocde/ford#2.00.00

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
FACULTAD DE INGENIERÍA INDUSTRIAL
UNIDAD DE POSGRADO

ACTA DE SUSTENTACIÓN N°035-UPG-FII-2026

SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE DOCTOR EN INGENIERÍA INDUSTRIAL

En la ciudad de Lima del día dos del mes de noviembre del año dos mil veintiséis, siendo las once horas, se instaló el Jurado Examinador para la Sustentación de la Tesis titulada: **"EVALUACIÓN DEL USO DE REDES SOCIALES Y SU IMPACTO SOBRE LA ELECCIÓN PRESIDENCIAL PERUANO 2021 MEDIANTE TÉCNICAS DE MACHINE LEARNING"**, presentado por el **Mg. EBER JOSEPH BALLON ALVAREZ,** para optar el Grado Académico de **DOCTOR EN INGENIERÍA INDUSTRIAL.**

Luego de la exposición y absueltas las preguntas del Jurado Examinador se procedió a la calificación individual y secreta, habiendo sido APROBADO con la calificación de DIECIOCHO (18) MUY BUENO.

El Jurado recomienda que la Facultad acuerde el otorgamiento del Grado Académico de **DOCTOR EN INGENIERÍA INDUSTRIAL** al **Mg. EBER JOSEPH BALLON ALVAREZ.**

En señal de conformidad, siendo las 12:05 **horas** se suscribe la presente acta en cuatro ejemplares, dándose por concluido el acto.

DR. FRANCISCO JAVIER WONG CABANILLAS
Presidente

DR. PEDRO PABLO ROSALES LOPEZ
Miembro

DR. EDUARDO WALTER BUSE THORNE
Miembro

DR. JOSÉ OVIDIO FLORES GUTIÉRREZ
Miembro

DR. JUAN MANUEL CEVALLOS AMPUERO
Asesor

UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

Anexo 1

CERTIFICADO DE SIMILITUD

Yo, **JUAN MANUEL CEVALLOS AMPUERO,** en mi condición de asesor(a) acreditado(a) mediante **DICTAMEN N°133-UPG-FII-2019** de la tesis, cuyo título es: **"EVALUACIÓN DEL USO DE REDES SOCIALES Y SU IMPACTO SOBRE LA ELECCIÓN PRESIDENCIAL PERUANO 2021 MEDIANTE TÉCNICAS DE MACHINE LEARNING",** presentado por el(la) magíster **EBER JOSEPH BALLON ALVAREZ** para optar el grado académico de **DOCTOR EN INGENIERÍA INDUSTRIAL**, **CERTIFICO** que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 05**%** de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado académico de **DOCTOR EN INGENIERÍA INDUSTRIAL.**

Lima, 01 de setiembre de 2026

UNMSM
Firmado digitalmente por CEVALLOS AMPUERO Juan Manuel FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 01.09.2026 09:19:27 -05:00

Firma del Asesor _____

Nombres y Apellidos del asesor: **JUAN MANUEL CEVALLOS AMPUERO**

DNI: **O7855059**

La vigencia del presente certificado corresponde a un período de noventa (90) días desde la fecha de su emisión; siempre que no haya observaciones en el texto durante el proceso de revisión por el jurado informante o surja alguna contingencia.

Dedicatoria

A Jehová Dios, mis queridos padres Manuel y Ceferina y en esta última etapa a mi esposa Saemi y mis queridas hijas Brunella, Doménica e Isabella, quienes han servido de inspiración y motivo para enfrentar esta etapa en mi desarrollo profesional. A mis queridas hermanas Marshury y Yesenia a quienes siempre llevo en mi corazón y pensamiento.

Agradecimientos

A mi asesor y profesores en el doctorado, colegas de la Universidad ESAN y colegas en la Universidad Nacional Mayor de San Marcos, quienes me impulsaron a seguir adelante en diferentes momentos en que parecía que la meta era inalcanzable.

Al PhD Javier del Carpio Gallegos, a quien desde que conocí en la Universidad Nacional Mayor de San Marcos como decano en la Facultad de Ingeniería Industrial y posteriormente en Universidad ESAN, como decano de la Facultad de Ingeniería, siendo directamente mi jefe, me ha servido de ejemplo para alcanzar los objetivos académicos y profesionales de forma responsable y ética.

A mis colegas y amigos especialistas en Ciencias de la Computación que contribuyeron con su revisión y comentarios sobre los diversos informes elaborados hasta la construcción del presente documento final.

Índice general

Acta de sustentación I
Dedicatoria II
Abstract XII
CAPÍTULO 1 INTRODUCCIÓN 1
1.1. Situación problemática 1
1.2. Formulación del problema 6
1.2.1. Esquema de promoción de las campañas electorales reguladas por ley 7
1.2.2. Problema general 1
1.2.3. Problemas específicos 2
1.3. Justificación de la investigación 2
1.3.1. Justificación teórica 2
1.3.2. Justificación práctica 3
1.4. Objetivos 5
1.4.1. Objetivo general 5
1.4.2. Objetivos específicos 5
CAPÍTULO 2 MARCO TEÓRICO 6
2.1. Marco Filosófico o epistemológico de la investigación 6
2.2. Antecedentes de investigación 9
2.3. Bases teóricas 16
2.3.1. Inteligencia artificial 16
2.3.2. Machine learning 20
2.3.3. Procesamiento de leguaje natural 27
2.3.4. Métodos de representación de texto 29
2.3.5. Algoritmos de clasificación 37
2.3.6. Métricas de evaluación 51
2.3.7. Herramientas y librerías 55
CAPÍTULO 3 METODOLOGÍA 60

3.1. Hipótesis general 60
3.2. Hipótesis específica 60
3.3. Diseño de la investigación 61
3.3.1. Diseño 61
3.3.2. Tipo 61
3.3.3. Enfoque 61
3.3.4. Población y muestra 61
3.3.5. Operacionalización de Variable 62
3.4. Tipo y diseño de investigación 69
3.4.1. Tipo de investigación 69
3.4.2. Enfoque 69
3.4.3. Población y muestra 69
3.4.4. Metodología de implementación 70
3.5. Aplicación metodológica 72
3.5.1. Extracción de datos 72
3.5.2. Análisis exploratorio y limpieza de datos 78
3.5.3. Vectorización de datos 81
3.5.4. Aplicación de transformadores y clasificación 89
3.5.5. Afinamiento 89
3.5.6. Evaluación y selección de modelo 89
CAPÍTULO 4 RESULTADOS Y DISCUSIÓN 98
4.1. Extracción y procesamiento de los datos 98
4.1.1. Debates presidenciales 98
4.1.2. Data procesada de Twitter 100
4.1.3. Clasificación de tópicos políticos 109
4.1.4. Evaluación 113
4.1.5. Configuración de los experimentos 117
4.1.6. SVM 118
4.1.7. Optimización de hiperparámetros 128
4.1.8. Análisis de resultados 129
4.2. Propuesta de framework para el tratamiento de FakeNews en procesos de elección de autoridades 139
4.2.1. Recolección de la fuente de datos 139

4.2.2. Procesamiento de los datos 140
4.2.3. Vectorización de los datos 140
4.2.4. “Liguistic Features” 142
4.2.5. Predicción 143
4.2.6. Métricas de evaluación 145
4.3. Aportes de la Tesis 146
CONCLUSIONES 160
RECOMENDACIONES 162
REFERENCIAS BIBLIOGRÁFICAS 164

Lista de tablas

Tabla 2.1: **Definiciones de IA organizadas en cuatro categorías.** 17
Tabla 2.2: Matriz term-documents basada en obras de Shakespeare 30
Tabla 3.1: Operacionalización de las variables 62
Tabla 4.1: Organización de tópicos en debates presidenciales 99
Tabla 4.2: Número de pseudo oraciones en transcripciones 99
Tabla 4.3: Tweets distribuidos por posición política, candidato y tópico 105
Tabla 4.4: Resumen de score kappa 108
Tabla 4.5: Relación de partidos considerados en la investigación 118
Tabla 4.6: Resultados de modelos F1Score (Macro y Micro) 130
Tabla 4.7: Resultados de los modelos en términos de F1 Score – Corpus Tweets 131
Tabla 4.8: Resultado de los modelos F1 score (Solo Tweets) 132
Tabla 4.9: Feature para el proceso experimental 143

Lista de figuras

Figura 1.1 Proceso general de elección electoral y gestión de la franja electoral.....9
Figura 1.2 Distribución del financiamiento público directo - Proceso electoral 2021....10
Figura 1.3 Reglamento de Financiamiento y Supervisión de fondos partidarios, artículos 6, 7 y 8.....1
Figura 2.1 Espectro para la representación de conocimiento y la capacidad de razonamiento. 8
Figura 2.2 Flujo de análisis de sentimiento.....9
Figura 2.3 Metodología.....11
Figura 2.4 Metodología para la detección de un social bot.....13
Figura 2.5 Representación de un agente en IA 21
Figura 2.6 Arquitectura de CBOW y Skip-gram en Word2Vec. 33
Figura 2.7 Clasificación de support vector machine.....38
Figura 2.8 (a) Hiperplano que separa a dos clases y (b) Hiperplano con margen máximo que separa a dos clases.....39
Figura 2.9 Arquitectura de la red convolucional LeNet.....41
Figura 2.10 CNN con aplicación en datos textuales. 44
Figura 2.11 Arquitectura de una RNN simple o Elman Network. 46
Figura 2.12 RNN para clasificación de textos. 47
Figura 2.13 Flujo de compuertas en una unidad neuronal de LSTM.....50
Figura 2.14 Matriz de Confusión de ejemplo. 52
Figura 3.1Framework metodológico.....70
Figura 3.2 Framework metodológico con detalle de técnicas y herramientas usadas 71
Figura 3.3 Algoritmo para la extracción de tweets de candidatos a la presidencia usando twint 73
Figura 3.4 Código Fuente para la extracción de tweets de candidatos a la presidencia usando twint 74
Figura 3.5 Proceso para el análisis y anotación de los tweets extraídos 75

Figura 3.6 Extracción de frases en textos de forma automática.....77
Figura 3.7 Procedimiento de limpieza para un tweet/frase de debate político.....80
Figura 3.8 Procedimiento de limpieza para una frase extraída de un plan de gobierno. 81
Figura 3.9 Proceso de vectorización para experimentos con Machine Learning.....84
Figura 3.10 Flujo del pipeline programado para la vectorización de datos en ML. 85
Figura 3.11 Proceso de vectorización para experimentos con Deep Learning. 86
Figura 3.12 Proceso de carga de datos y fine-tuning al modelo de words embedding. ..88
Figura 3.13 Proceso de entrenamiento, fine-tuning y evaluación de los modelos. 92
Figura 3.14 Flujo del pipeline programado para clasificación usando técnicas de Machine Learining. 93
Figura 3.15 Flujo del modulo programado para la clasificación usando técnicas de Deep Learning. 94
Figura 3.16 Proceso de selección del mejor modelo. 96
Figura 4.1 Distribución de tópicos en porcentaje. 101
Figura 4.2 Distribución de tweets por tópicos – Paridos de “La derecha” 102
Figura 4.3 Distribución de tweets por tópicos – Paridos de “La izquierda” 103
Figura 4.4 Distribución de tópicos por candidato. 106
Figura 4.5 Distribución de tópicos por candidato. 107
Figura 4.6 Distribución de tokens en corpus final. 115
Figura 4.7 Distribución de categorías en corpus final en porcentaje. 116
Figura 4.8 Esquema cross-validation. 120
Figura 4.9 Flujo del modelo SVM 120
Figura 4.10 Arquitectura propuesta por Bilbao & Jayo. 123
Figura 4.11Arquitectura simplificada tomando como referencia la arquitectura original propuesta por Bilbao & Jayo. 125
Figura 4.12 Arquitectura basada en RNN. 127
Figura 4.13 Curva de entrenaniento y loss de modelo CNN. 134
Figura 4.14 Curva de entrenamiento y loss del modelo CNN. 135
Figura 4.15 Curvas de entrenamiento y loss del modelo RNN. 136
Figura 4.16 Modelos de vectorización de texto divididos en grupo. 142
Figura 4.17 Arquitectura referencial con capas convolucionales 145

Figura 4.18 Arquitectura referencial con capas recurrentes.....145

Resumen

La presente investigación analiza instrumentos digitales difundidos a través de redes sociales, planes de gobierno y transcripciones de debates, mediante técnicas de Procesamiento de Lenguaje Natural (PLN) complementadas con métodos de Machine Learning, en el contexto de la elección presidencial peruana de 2021. Dicho proceso electoral se desarrolló bajo condiciones particulares: en primer lugar, la influencia del contexto denominado "primavera latinoamericana", caracterizado por una mayor presencia de partidos de izquierda en los gobiernos de la región; en segundo lugar, el impacto del periodo pandémico iniciado en 2020, que reconfiguró las estrategias de proselitismo político y posicionó a las redes sociales como un instrumento central para la difusión de propuestas; y, en tercer lugar, la necesidad de proponer un framework metodológico que permita replicar el análisis en futuros procesos electorales.En este marco, la investigación propone un framework metodológico orientado al análisis y procesamiento de tres fuentes textuales principales, con énfasis en su replicabilidad y aplicabilidad en distintos contextos electorales.

Los resultados obtenidos evidencian que las métricas accuracy, F1-score micro y F1-score presentan un desempeño eficiente para la evaluación de modelos predictivos en tareas de clasificación.

Finalmente, el framework propuesto y validado se consolida como un marco de referencia para el análisis de futuros procesos electorales, considerando que, desde 2015, las redes sociales y los instrumentos digitales se han convertido en componentes fundamentales de las estrategias proselitistas en las campañas políticas.

Palabras clave: Procesamiento de lenguaje natural, Computación liguistica, Machine Learning, Deep Learning, Manifesto Project, Texto político

Abstract

This research analyzes digital instruments disseminated through social media, government plans, and debate transcripts, using Natural Language Processing (NLP) techniques complemented by Machine Learning methods, in the context of the 2021 Peruvian presidential election. This electoral process unfolded under particular conditions: first, the influence of the so-called “Latin American Spring,” characterized by a greater presence of left-wing parties in the region's governments; second, the impact of the pandemic period that began in 2020, which reconfigured political campaigning strategies and positioned social media as a central instrument for disseminating proposals; and third, the need to propose a methodological framework that allows the analysis to be replicated in future electoral processes. Within this framework, the research proposes a methodological framework oriented toward the analysis and processing of three main textual sources, with an emphasis on its replicability and applicability in different electoral contexts.

The results obtained demonstrate that the accuracy, F1-score micro, and F1-score metrics perform efficiently for evaluating predictive models in classification tasks.

Finally, the proposed and validated framework is consolidated as a reference framework for analyzing future electoral processes, considering that, since 2015, social media and digital tools have become fundamental components of campaign strategies in political campaigns.

Keywords: Natural language processing, Computational liguistic, Machine Learning, Deep Learning, Manifesto Project, Political text.

CAPÍTULO 1 INTRODUCCIÓN

1.1. Situación problemática

El año 2020 (a raíz de la pandemia por el COVID-19), es considerado un año que marcará un antes y después, por el impacto global, esto, debido a las diversas restricciones que se impuso por cada país. Frentes como el económico, social, regional y sobre todo en la salud. Por ejemplo, de acuerdo con CCSA et al. (2020) los bancos franceses, japoneses, británicos y estadounidenses, todos con activos extranjeros totales sobre EMDE superiores a 450.000 millones de dólares, registraron cerca o menos del 50% de sus posiciones a nivel local en el país prestatario. En lo relacionado a la empleabilidad, se observó un decremento 10.5%, lo que equivale a 305 millones de trabajadores a tiempo completo con una semana laboral de 48 horas.

Previo a las elecciones presidenciales de 2021, resulta pertinente destacar los procesos electorales desarrollados en la región durante 2019 y 2020, en los cuales se eligieron gobiernos de orientación socialista o de izquierda. En Argentina, el 27 de octubre de 2019 fue electo **Alberto Fernández**. En Uruguay, el 24 de noviembre de 2019 se celebraron elecciones en las que resultó ganador **Luis Lacalle Pou**. En Bolivia, las elecciones del 20 de octubre de 2019 dieron inicialmente como ganador al candidato oficialista, aunque dicho proceso fue posteriormente anulado. Finalmente, en los comicios del 18 de octubre de 2020, Bolivia eligió a **Luis Arce** como presidente.

En el Perú, el 6 de junio del 2021, se tuvo programado las elecciones presidenciales en un contexto restrictivo por las directrices sanitarias que limitaron el desarrollo regular

de las campañas de los partidos contendientes que por lo regular se caracterizaba por convocatorias masivas de la ciudadanía en lugares públicos.

Un aspecto relevante en el proceso de transmisión de información es la estrategia utilizada en la comunicación, que en el caso de la interacción humana resulta ser indispensable para su superviviencia. Siendo relevante la comunicación, el ser humano ha utilizado los recursos de momento o de la época para transmitir sus mensajes, tales como el papel, la radio, la televisión y desde 1985 internet. Junto con el robustecimiento de las comunicaciones a través de internet, en 1997 nace la primera red social, denominada SixDegrees, red social que se enfoca en resaltar que cualquier persona se encuentra a solo seis pasos o círculos sociales de la persona más distante. Posteriormente, surgen redes sociales como Hi5 o MySpace que paulatinamente fueron desplazadas por Facebook y Twitter. De acuerdo a lo expuesto por Khatua et al. (2020), Twitter, se convirtió para los políticos como el medio más importante a la hora de tratar de comunicar sus mensajes. Para la elección presidencial del 2016 en los Estados Unidos de América los 17 candidatos en competencia, de alguna manera utilizaron la mencionada red social, donde Donald Trump contó con 30 millones de seguidores, mientras que su directa rival contaba con 15 millones. De acuerdo con Flamino et al. (2021), cuando Joe Biden fue electo como presidente de Estados Unidos, el uso de las redes sociales tuvo una relevancia notable en los procesos electorales de 2016 y 2021, los autores subrayan la presencia de una fuerte polarización, ya que los ciudadanos tienden a consumir información proveniente de fuentes alineadas con su propia orientación política. Es importante destacar que, en el caso de las redes sociales, no solo influye el consumo de información, sino también la práctica de republicarla, esta dinámica amplificó la distancia entre las comunidades electorales en conflicto. Un elemento relevante del proceso electoral de 2020 fue la consolidación de los influencers noticiosos, cuyo uso intensivo de las plataformas digitales impulsó incluso que los candidatos recurrieran a ellos para promover sus propuestas políticas.

De acuerdo con (Fenoll el al., 2023), al igual que en Estados Unidos, las elecciones celebradas en América Latina entre 2019 y 2023 se caracterizaron por una marcada polarización del discurso político derivada del uso de las redes sociales. Un aspecto diferenciador es que, además de estas plataformas, también se difundió información a

través de WhatsApp. Asimismo, durante ese mismo periodo, TikTok se consolidó en la región como un medio de comunicación relevante; sin embargo, es importante considerar que esta red social se basa principalmente en contenido audiovisual en formato de video.

Un primer componente a considerar, en el contexto de la presente investigación es la red social, que desde la aparición de internet ha sufrido variantes en cuanto a su definición; vista desde la perspectiva tradicional, presenta el siguiente significado: "Constitución de grupos o nodos que implica el establecimiento de una red en el que se establecen vínculos (unión entre nodos) y que tienen determinadas repercusiones entre ellos, producto de su interacción" (Requena, 1989, p. 148). Esta definición, evolucionó, gracias a la mejora de los servicios de internet, fundamentalmente desde la aparición de redes sociales como Facebook o Twitter. De acuerdo con (Torres, 2009, p. 36) significa "Sitios web que permiten la creación y el intercambio de información y contenidos por las personas y para las personas, en las que el consumidor es a la vez productor y consumidor de la información. Tales redes reciben ese nombre porque son sociales, libres y abiertas para la colaboración e interacción de todos, y porque son medios, es decir, medios de transformación de información y contenidos". Este componente resulta importante en vista que dado el confinamiento al que se estuvo sometido por la pandemia, los candidatos y/o partidos políticos utilizaron este medio como alternativa de comunicación.

Un segundo componente está vinculado al procesamiento de datos de los planes de gobierno que todo partido político expone como parte de su estrategia de campaña, que en caso el candidato del mencionado partido salga elegido aplicará como referencia a la hora de gobernar. El componente es relevante porque se persigue aplicar un método de procesamiento para la identificación de tópicos clave.

Un tercer componente en la investigación está ligado al procesamiento de la transcripción de los debates presidenciales que de forma regular se realiza de forma oral. El componente es relevante porque se persigue aplicar un método de procesamiento para la identificación de tópicos clave al igual que en componente anterior.

A continuación, se destacarán antecedentes relativos a cómo el uso de las redes sociales han sido relevantes en hechos considerados críticos por su alcance regional que en ocasiones ha implicado el desarrollo de conflictos bélicos lo que permite catalogar la investigación a desarrollar como relevante.

El 24 de junio del 2014 Al Bagdadi declara desde Mosul el nacimiento del autodenominado ESTADO ISLÁMICO que previamente tenía la denominación de ISIS (Islamic State of Iraq and Syria), esto en apoyo a un grupo de insurgentes cuya lucha era la del retiro de tropas estadounidenses de Iraq. De acuerdo con (Campbell, 2019) el número de militantes extranjeros implicados en el conflicto fue de alrededor 40000 miembros. El reclutamiento de estos militantes extranjeros se realizó mediante el intenso uso de redes sociales. De acuerdo con (El Ghamari, 2017) fue a partir del 2010 en que los grupos jihadistas empezaron a hacer uso fundamentalmente de la red soscial Twitter, esto para la promoción de sus actividades. En primera instancia los mensajes promocionaban diversos foros de discusión y/o promoción de su pensamiento, pero posteriormente miembros entusiastas del mismo grupo no solo empezaron a replicar los Twitters sino que empezaron a promover su propio contenido y al mismo tiempo a invitar a nuevos miembros. Para el CSCC (Department's Center for Strategic Counterterrorism Communications) la mitad de la lucha contra este tipo de grupos se debería realizar en las redes sociales, lo cual evidencia que este contexto a la fecha resulta extremadamente crítico.

Un hecho trascendental que destaca la importancia de las redes sociales en las contiendas políticas es lo acontecido en la elección presidencial de Estados Unidos en el 2016 el cual estuvo caracterizado por el uso intenso de las redes sociales por el outsider de momento, Donald Trump. En la etapa más crítica de la campaña Hillary Clinton hacia un uso bastante formal de las redes sociales mientras que Donald Trump hacía uso informal (fundamentalmente twitter) lo que atrajo la atención de asiduos consumidores de este tipo de servicio, esto se reflejaba en la interacción posterior a los tweets los cuales eran replicados. De acuerdo con (Gottfried, 2016), para el año en cuestión el 62% de los americanos adultos hacía consumo de las noticias que circulaban en los denominados social media. Al margen del uso de las redes sociales, hubo hechos cuestionables en el uso

de las redes sociales en plena campaña que al final resultaron críticos en la elección del nuevo presidente y era que mucha de la "información" que circulaba era catalogada como falsa (fake news). De acuerdo con (Silverman, 2016) las noticias falsas fueron ampliamente más distribuidas por redes sociales en relación a noticias formales generadas en diarios formales como el New York Times o Washington Post. En los tres meses altamente críticos, 20 historias falsas acerca de las elecciones, generaron alto rendimiento para sitios de dudosa procedencia los cuales generaron 8,711,000 acciones o reacciones y comentarios en Facebook. A este respecto (Parkinson, 2016) defiende la posición que uno de los factores críticos de la elección del presidente Trump fueron las noticias falsas.

Un evento de corte tecnológico relevante que evidencia el rol que hoy juegan las redes sociales y su mal uso es lo realizado por la compañía Cambridge Analytica (CA). Esta compañía fue originalmente creada para brindar servicios de minería de datos pero luego pasaría a ser una compañía para la comunicación estratégica para procesos electorales. De acuerdo con lo divulgado en su momento en la página web de CA, trabajaron en más de 100 campañas electorales en diversos países lo cual no tendría nada sorprendente debido a que muchas compañías brindan este tipo de asesoría. Lo controversial acerca del accionar de la compañía es el uso de miles de perfiles de usuarios de Facebook recopilados de forma indebida. El escándalo se da inicio en el 2013 con una actividad aparentemente regular impulsada por el científico de datos Aleksandr Kogan, quien desarrolló una encuesta en la plataforma de Facebook relativo a un test de personalidad. A lo largo del tiempo llegó a recopilar la información directa de 265000 usuarios pero sin que estén enterados, la aplicación posibilitó que se accediera a la red de amigos de los usuarios que aplicaron al test. Al final de este experimento Kogan llegó a recopilar la información de 50 millones de personas (15% de la población estadounidense). La información recopilada incluía desde simples likes hasta comentarios de la red de contactos de los 265000 usuarios preliminares. Luego de realizar esta recopilación Kogan vendió esta información a Cambridge Analytica. Uno de los hechos resaltantes es que CA, cruzó los datos del test de Kogan con la información de Facebook para realizar clusteres de usuarios. Una vez identificado los clusteres CA diseñó y desarrolló noticias falsas ajustadas a cada cluster para inclinar el voto. En un proceso de

investigación encubierto a Alex Tayler (jefe de datos de CA), este declaró lo siguiente: "Si estás recolectando información de personas y estás haciendo un perfil de ellos, eso te da más conocimiento que puedes usar para saber cómo segmentar la población para darles mensajes sobre temas que les importan, y usar un lenguaje e imágenes con los que es probable que se involucren". El escándalo de CA alcanzó al mismo Facebook sobre cómo es que la red social hacía mínimos esfuerzos para evitar la propagación de noticias falsas.

Finalmente, un aspecto relevante en la investigación es el procesamiento de los planes de gobierno de cada partido contendiente, así como los debates presidenciales. Tanto los planes de gobierno como los debates (en especial este último) resultan ser sumamente valiosos porque muchos ciudadanos indecisos se inclinan por uno u otro candidato. Respecto a este aspecto, se destaca el proyecto "Manifesto" (Manifesto, 2021) que desde el 2009, analiza los programas electorales de los partidos para estudiar las preferencias políticas de los partidos donde puntualmente se realiza la recopilación y análisis comparativo del contenido de los manifiestos (planes de gobierno) de los partidos con el apoyo de codificadores de más de 50 países diferentes utilizando técnicas de Procesamiento de Lenguaje Natural, esto, de acuerdo con lo señalado por (Ares & Volkens 2017).

Por lo expuesto hasta este punto la investigación a desarrollar se concentrará en el estudio de la repercusión del uso de las redes sociales, análisis de planes de gobierno y el análisis de la transcripción de los debates presidenciales en el contexto de la elección presidencial peruano en el 2021.

1.2. Formulación del problema

Un primer aspecto relativo a la problemática a estudiar es la carencia de un marco de referencia (Framework) para entender la problemática. De acuerdo con (Vinz, 2015) se destaca el hecho de que uno de los objetivos de los investigadores es la presentación de teorías para la explicación de fenómenos, establecer conexiones y hacer predicciones, para esto los investigadores postulan o desarrollan marcos de referencia para el entendimiento de problemáticas. El objetivo de un marco de referencia es la presentación formal y

estructurada de un conjunto de procesos para la explicación y/o el seguimiento de un conjunto de actividades para alcanzar un determinado objetivo. Para que el marco de referencia quede formalizado y aceptado por la comunidad debe pasar por procesos de validación utilizando métodos formales soportado por procesos estadísticos (equivalentes) o por juicio de expertos.

Para la investigación, el marco de referencia resulta relevante pues implica el diseño de un conjunto de actividades que abarque el proceso de extracción de grandes volúmenes de datos gestados a través de las redes sociales, el tratamiento de data no estructurada expresada en lenguaje natural, el proceso de clasificación y posterior interpretación y determinación de los efectos en el proceso de elección presidencial. Según (Jiménez, 2019) destaca la propuesta de un framework para el proceso de análisis sentimental de data extraída desde redes sociales lo cual implica la definición de una variedad de etapas para el tratamiento del procesamiento de lenguaje natural.

Un segundo aspecto es la identificación de eventos categorizados como críticos que incidan sobre el proceso de elección presidencial. La identificación del evento crítico resulta problemático debido a que la información expuesta en redes sociales es de corte no estructurada lo que amerita la aplicación de una metodología basada en procesamiento de lenguaje natural.

Finalmente, el tercer aspecto de la investigación, es la ideintificación de un mecanismo que posibilite la categorización de lo publicado en redes sociales en las campañas electorales por procesos de entrenamiento.

1.2.1. Esquema de promoción de las campañas electorales reguladas por ley

De acuerdo con el Jurado Nacional Electoral el proceso electoral esta constituido por un conjunto de etapas que están amparadas en la ley, para precisar, en el artículo 79° de la Ley Orgánica de Elecciones.

Los organismos que participan en la gestión de todo proceso electoral son:

- Jurado Nacional Electoral (JNE)
- Oficina Nacional de Procesos Electorales (ONPE)

- Registro Nacional de Identificación y Estado Civil (RENIEC)

El financiamiento de las campañas políticas en el Perú, pueden provenir de fuentes públicas y privadas. El proceso de financiamiento público de las campañas está regulado por la Ley de Partidos Políticos, N° 28094. La responsabilidad de efectuar la verificación y control de esta actividad recae en la ONPE, esto, de acuerdo a lo expuesto en el Reglamento de Financiamiento y Supervisión de Fondos Partidiarios de la Oficina Nacional de Procesos Electorales (Diario Oficial El Peruano, 2025)

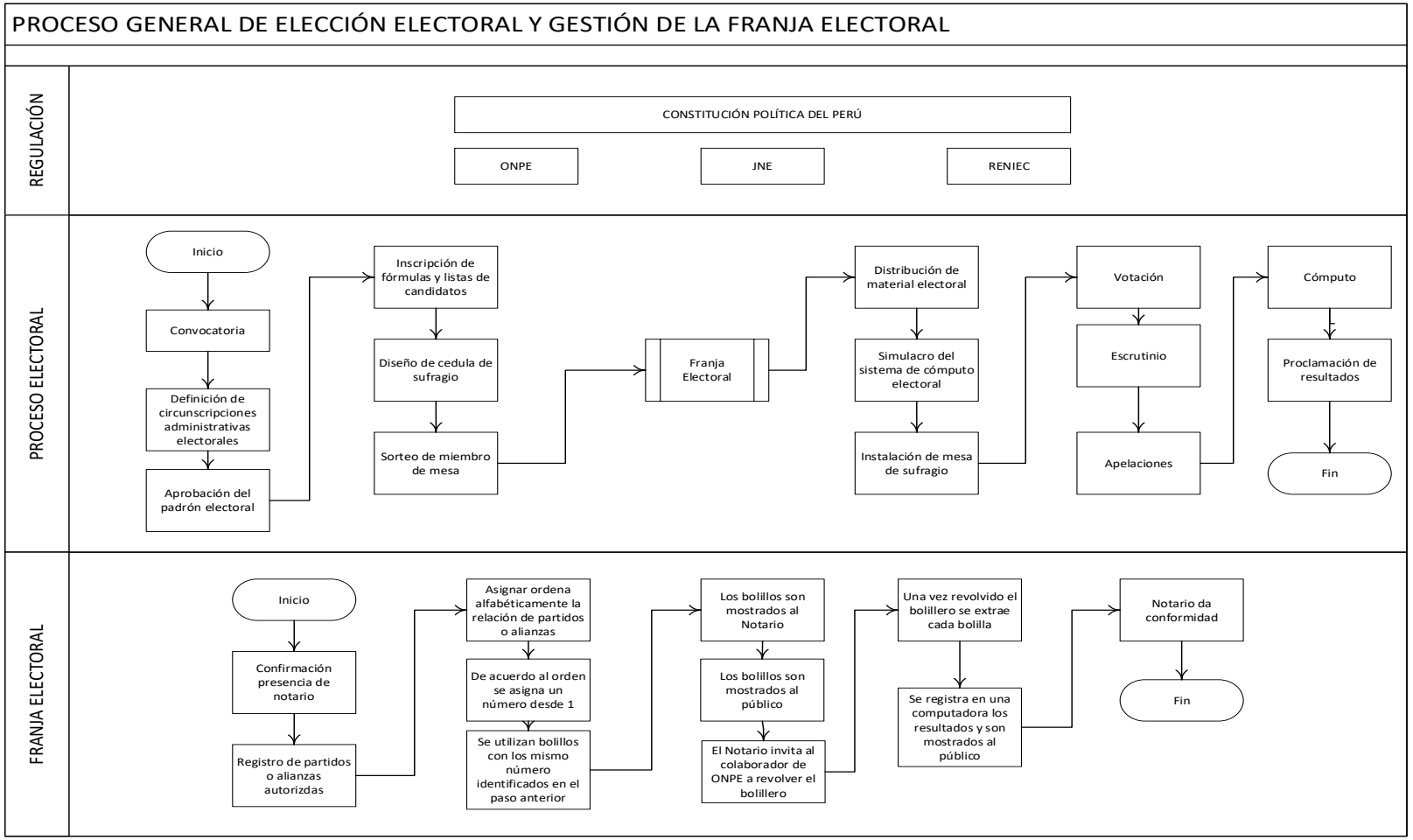
El financiamento de las campañas políticas en el Perú, se da fundamentalmente para poder hacer uso, de la denominada Franja Electoral, en el que la ONPE dosifica de manera justa el tiempo en el que los contendientes harán uso de los principales medios de comunicación para la divulgación de sus propuestas políticas.

Adicionalmente, independientemente del tipo de elección, la campaña implicará la transmisión de propuestas, usando el leguaje natural, a través del discurso proselitista, con el objetivo de inclinar o inducir al voto hacia el postulante y/o hacer declinar el voto hacia el contendiente.

En la figura 1.1 se detalla, mediante un diagrama de flujo la configuración deel proceso electoral, destacando la franja electoral. En la figura 1.2 se detalla el financiamiento público directo que recibió cada partido para el desarrollo de su campaña electoral 2021.

En la figura 1.3 se detalla, el esquema de Reglamento de Financiamiento y Supervisión de fondos partidarios.

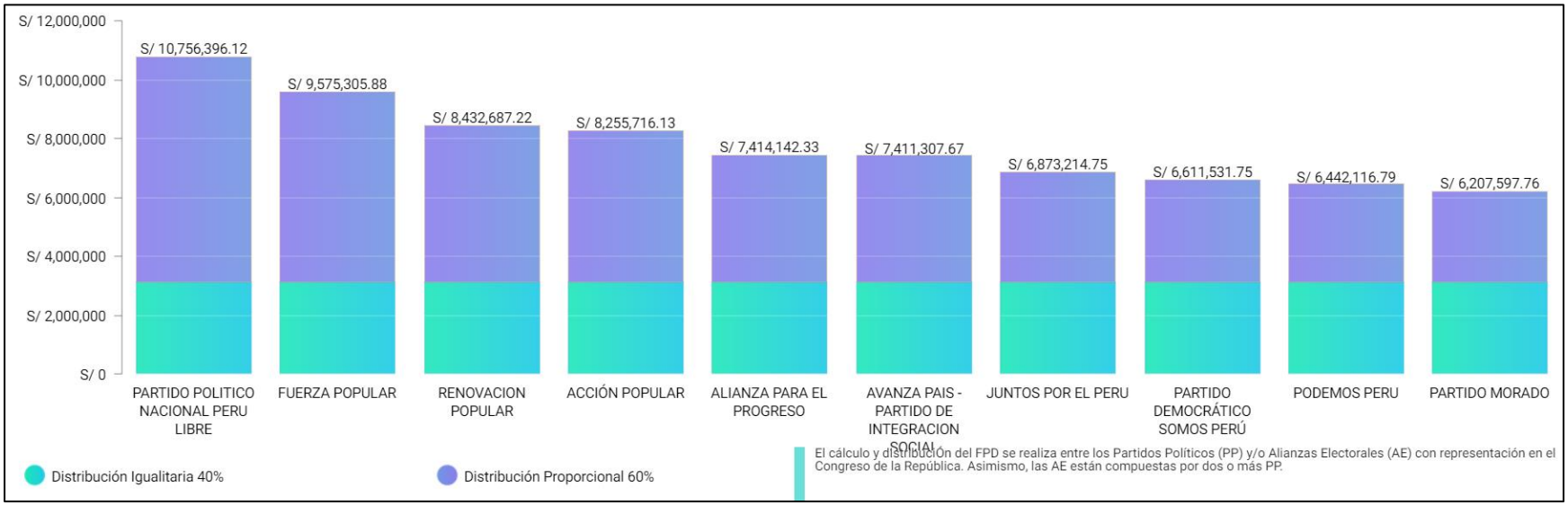
Figura 1.1 Proceso general de elección electoral y gestión de la franja electoral



Fuente: Adaptado de informe: "Etapas del proceso electoral" (Gobierno Peruano, 2024)

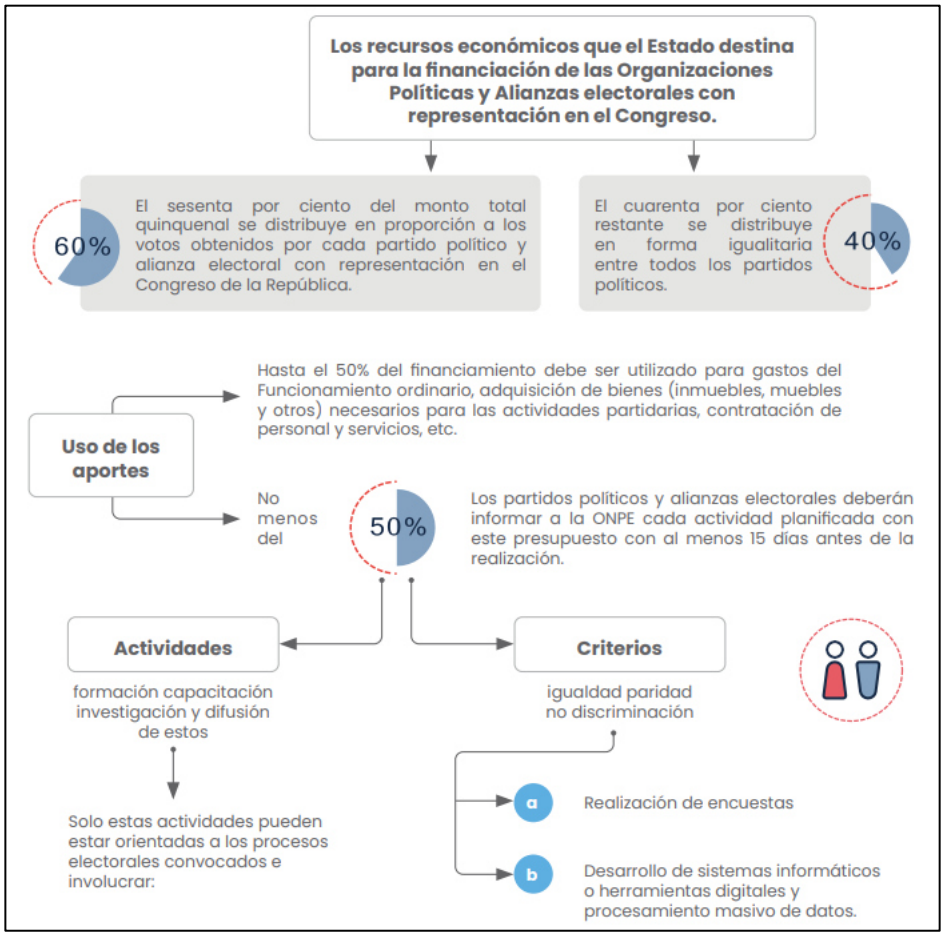
Nota: El diagrama muestra el proceso electoral en Perú regulado por la Constitución y organismos como ONPE, JNE y RENIEC. Incluye etapas: convocatoria, definición de circunscripciones, padrón electoral, inscripción de candidatos, diseño de cédulas, sorteo e instalación de mesas, votación, escrutinio y cómputo de resultados. Además, detalla la gestión de la franja electoral, donde se sortean y registran los turnos de participación de los partidos con supervisión notarial hasta su publicación.

Figura 1.2 Distribución del financiamiento público directo - Proceso electoral 2021



Fuente: Adaptado de Gobierno Peruano, (2021). Distribución Financiamiento Público Elecciones Presidenciales 2021
Nota: El gráfico muestra la distribución del financiamiento público directo a partidos políticos en Perú, dividido en 40% igualitario y 60% proporcional según representación en el Congreso. Los partidos con mayor monto asignado son Perú Libre y Fuerza Popular. Los montos disminuyen progresivamente para otros partidos como Renovación Popular, Acción Popular y Partido Morado según su representación parlamentaria.

Figura 1.3 Reglamento de Financiamiento y Supervisión de fondos partidarios, artículos 6, 7 y 8.



Fuente: Adaptado de Gobierno Peruano (2021). Reglamento de Financiamiento y Supervisión de Fondos Partidarios.
Nota: El Estado financia a los partidos políticos con representación en el Congreso: 60% del monto se distribuye proporcionalmente según votos y 40% de forma igualitaria. Hasta 50% del financiamiento puede usarse para funcionamiento y bienes, y al menos 50% para actividades de formación, capacitación, investigación y difusión. Los partidos deben informar a la ONPE sobre sus actividades y cumplir criterios de igualdad, paridad y no discriminación.

En este acápite se observa que no existe un mecanismo regulado para la gestión de la información publicada en redes sociales, siendo estas últimas de alto impacto dado el grado de penetración estas últimas en la población.

1.2.2. Problema general

¿De qué manera el desarrollo de un framework basado en técnicas de Procesamiento de Lenguaje Natural y Machine Learning posibilitará la identificación y el tratamiento de tópicos de interés, así como la clasificación de lo publicado en redes sociales, planes de gobierno de partidos políticos y lo expuesto en el debate presidencial, para realizar procesos de predicción y clasificación de posts en el contexto de la elección presidencial de 2021?

1.2.3. Problemas específicos

PE1: ¿De qué manera la propuesta de un framework para el análisis de lo expresado en redes sociales, planes de gobierno y debate presidencial, posibilitará servir de referencia para procesos de clasificación y predicción de post en el contexto de la elección presidencial 2021 que incluyan técnicas de Procesamiento Natural y técnicas de Machine Learning?

PE2: ¿Qué técnicas de Procesamiento de Lenguaje Natural posibilitará el procesamiento de posteos en redes sociales, análisis de planes de gobierno y transcripciones del debate presidencial 2021?

PE3: ¿Qué técnicas de Machine Learning serán las pertinentes para el procesamiento de tópicos de interés de data proveniente de redes sociales, planes de gobierno y el debate presidencial 2021?

PE4: ¿Qué algoritmo de Machine Learning y respectivamente métricas de evaluación permitirá la predicción de clasificación de post de proselitismo político en redes sociales?

1.3. Justificación de la investigación

1.3.1. Justificación teórica

Para el 2021, de acuerdo a lo reportado por OSIPTEL la velocidad promedio de descarga en Perú aumentó 20.6%, con dicha mejora, el Perú se posicionó en el 6to lugar a nivel sudamericano. La cobertura de internet a nivel de hogares fue de 87%. En relación a la inclusión sobre el uso de internet, de acuerdo a la revista especializada Insigth (2021), Perú tiene la calificación de 70.3/100 puntos, lo que indica una buena penetración del

servicio a poblaciones diversas. El acceso generalizado a internet en el territorio nacional ha propiciado la consolidación de las redes sociales como espacios sociotécnicos de interacción, en los cuales los ciudadanos no solo acceden a información de interés público, sino que también participan activamente en la construcción del discurso social mediante la expresión de opiniones, valoraciones y posicionamientos en torno a diversas problemáticas de carácter social y político. Estas plataformas digitales se constituyen, por tanto, en fuentes relevantes para el análisis del comportamiento discursivo y de la percepción ciudadana en contextos contemporáneos.

Es importante señalar que para el año 2021 TikTok ya tenía una presencia destacada en el ecosistema de redes sociales, impulsada en gran medida por la pandemia. En comparación con Twitter, utilizada ampliamente en campañas políticas, la diferencia principal radica en el tipo de contenido y su dinámica de uso: mientras TikTok se caracteriza por ofrecer videos cortos, creativos, musicales y en formato vertical, sustentados en un algoritmo de recomendación altamente personalizado, Twitter se centra en el texto, la difusión de noticias en tiempo real y las conversaciones rápidas.

En este sentido, la información generada y difundida a través de las redes sociales (instrumento 1), los planes de gobierno de los partidos políticos en contienda (instrumento 2) y las intervenciones de los candidatos en los debates presidenciales (instrumento 3) presenta una naturaleza predominantemente no estructurada, caracterizada por alta variabilidad semántica, heterogeneidad discursiva y ausencia de una organización formal explícita. Dicha condición plantea la necesidad teórica y metodológica de desarrollar un enfoque analítico que permita identificar de manera sistemática los tópicos de interés presentes en cada uno de los instrumentos considerados y, de forma fundamental, evaluar el grado de coherencia discursiva existente entre ellos. Este análisis resulta central para comprender las relaciones entre el discurso político institucional y las manifestaciones discursivas ciudadanas, así como para aportar evidencia empírica sustentada en métodos computacionales al estudio de la comunicación política contemporánea.

1.3.2. Justificación práctica

En vista que la recolección de datos se realizará a partir de plataformas de redes sociales, el volumen, la heterogeneidad y la naturaleza predominantemente no estructurada de la información generada plantean desafíos significativos en términos de almacenamiento, escalabilidad y capacidad computacional para su procesamiento. En este contexto, las técnicas avanzadas de Aprendizaje Automático (Machine Learning) y Procesamiento del Lenguaje Natural (PLN) se consolidan como enfoques metodológicos maduros y robustos para el análisis de grandes volúmenes de texto, particularmente en escenarios caracterizados por alta complejidad y sensibilidad.

El PLN permite abordar el análisis semántico del lenguaje mediante modelos de representación del significado, tales como embeddings léxicos y modelos de lenguaje contextualizados, los cuales capturan relaciones semánticas, sintácticas y pragmáticas a partir del uso contextual de las palabras. Complementariamente, las técnicas de Machine Learning facilitan la identificación de patrones latentes, regularidades y estructuras subyacentes en los datos, superando las limitaciones de los enfoques basados exclusivamente en reglas o análisis manual.

La integración de técnicas del Machine Learning y PLN, articulada a través del framework propuesto, posibilitará el desarrollo de un modelo computacional capaz de abordar un escenario problemático de alta sensibilidad y complejidad, favoreciendo la extracción de conocimiento relevante a partir de información implícita, no observable directamente y dispersa en grandes volúmenes de texto. De este modo, el enfoque propuesto contribuye a una comprensión más profunda, sistemática y escalable de la problemática objeto de estudio, sustentada en principios de análisis semántico, aprendizaje estadístico y modelado computacional del lenguaje natural.

1.4. Objetivos

1.4.1. Objetivo general

Proponer un framework basado en técnicas Procesamiento de Lenguaje Natural y Machine Learning, para la identificación, tratamiento de tópicos de interés y clasificación en lo publicado en redes sociales, planes de gobierno de partidos políticos y lo expuesto en el debate presidencial para procesos de predicción de clasificación en el contexto de elección presidencial 2021.

1.4.2. Objetivos específicos

OE1: Proponer un framework que posibilite analizar lo expresado en redes sociales, planes de gobierno y debate presidencial, para usarlo como de referencia para procesos de clasificación y predicción de posts en el contexto de la elección presidencial 2021, mediante técnicas de Procesamiento de Lenguaje Natural y Machine Learning.

OE2: Identificar y seleccionar las técnicas de Procesamiento de Lenguaje Natural que posibilite el procesamiento de posteos en redes sociales, análisis de planes de gobierno y transcripciones para el análisis exploratorio del debate presidencial 2021.

OE3: Identificar y seleccionar las técnicas de Machine Learning para el procesamiento de los tópicos de interés de data proveniente de redes sociales, planes de gobierno y transcripción del debate presidencial 2021.

OE4: Identificar y seleccionar el algoritmo de Machine Learning y respectivamente las métricas de evaluación que permita predecir la clasificación de posts de proselitismo político de redes sociales.

CAPÍTULO 2 MARCO TEÓRICO

2.1. Marco Filosófico o epistemológico de la investigación

Jean Piaget, considerado el padre de la epistemología, a raíz de su actividad docente con niños en Francia, destaca el proceso de desarrollo cognitivo del ser humano el cual es gradual, resalta el hecho de que mientras va pasando el tiempo, gracias al uso de los recursos sensoriales y su interacción con el entorno (entrenamiento), este va adquiriendo mayor conocimiento. Esto permite inferir que un adulto tendrá mayores habilidades cognitivas en relación a un niño. Gracias a sus investigaciones es que surge la teoría del constructivismo del aprendizaje y como consecuencia conocimiento. Esto se ve complementado con la habilidad del ser humano de poder retener el conocimiento a través de la escritura lo cual posibilita la representación del conocimiento a través de símbolos.

Parte de las actividades que todo investigador realiza y que tiene como marco metodológico la epistemología es el establecimiento de relaciones epistémicas, sobre todo aquellas que permiten determinar relaciones de inferencia para dar sustento al conocimiento. El esfuerzo de identificar relaciones de inferencia se da desde el 384 A.C. con Aristóteles con quien nace la lógica a través de los silogismos. A raíz de la emergencia de la lógica es que se sientan una de las bases fundamentales de la inteligencia artificial el cual es la emulación de la capacidad de aprendizaje que tiene el ser humano pero lo destacable es su capacidad para representar el conocimiento a través de reglas de inferencia.

De acuerdo con (Wheeler, 2004) los epistemólogos se esfuerzan en incorporar definiciones formales para nuevos fenómenos como parte de sus teorías de definición de

nuevo conocimiento, pero también es ideal que estos mismos epistemólogos atiendan contextos de representación de conocimiento donde la inteligencia artificial ha madurado, especialmente en la capacidad de representación del conocimiento con data estructurada e inclusive con data no estructurada, esto a través de técnicas como el Natural Language Processing y técnicas de Machine Learning (y variantes)

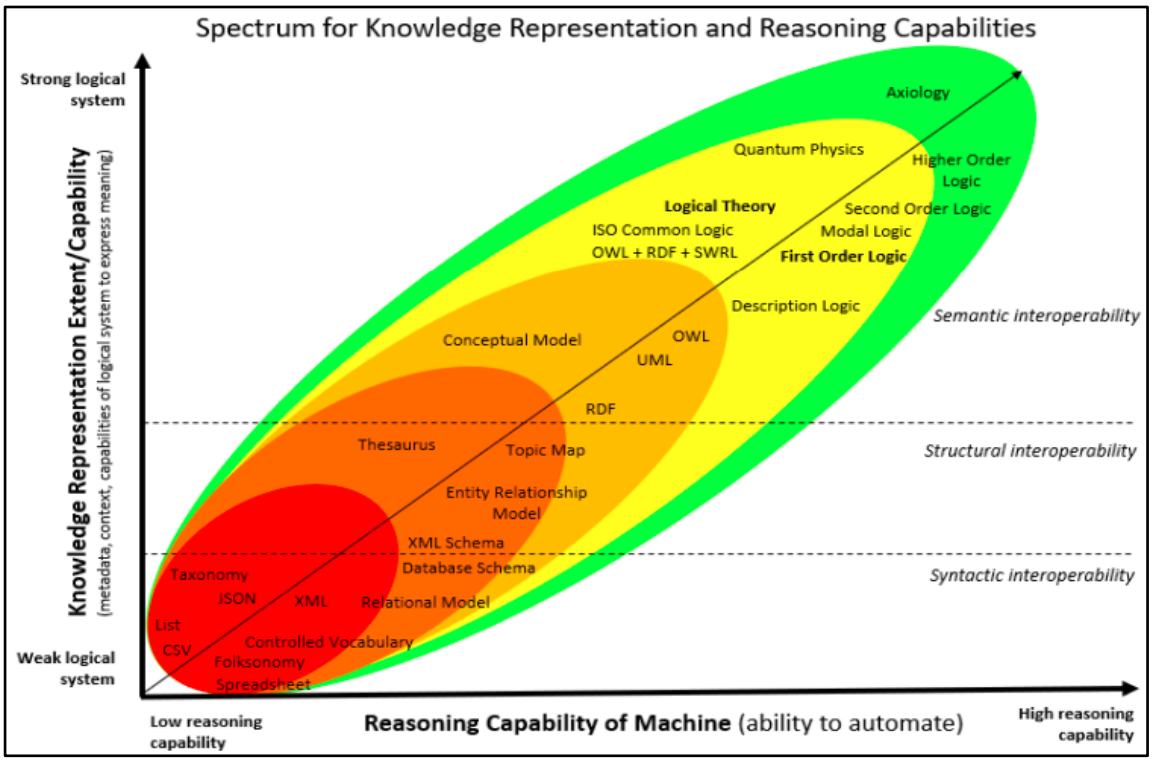
La importancia de esta investigación, desde la perspectiva de la Epistemológica se puede identificarla desde la descripción de la forma que en que los sistemas artificiales aprenden, lo "cuestionable" en este punto es que la actividad no está supeditada a la incorporación de recursos cognitivos propios de los seres humanos y carentes en los sistemas artificiales, por lo menos no hasta el día de hoy. Lo destacable es que tanto filósofos como científicos al haber entendido como es que el ser humano aprende y genera y acumula conocimiento para generar nuevo conocimiento han logrado establecer reglas o procedimientos para procesos de aprendizaje e inferencia, son estas reglas y el desarrollo de la capacidad computacional lo que ha posibilitado emular de alguna manera como es que el ser humano aprende.

De acuerdo a la RAE, se define el término ontología como: "ciencias de la comunicación y en inteligencia artificial, red o sistema de datos que define las relaciones existentes entre los conceptos de un dominio o área del conocimiento" Son precisamente los ontólogos quienes se esfuerzan en los procesos de clasificación de conceptos a través de sus relaciones. La ontología resulta relevante, en el contexto epistemológico, debido a que se hace un escuerzo para la representación de la información. Kendall y McGuiness (2019) proponen una variación en cuanto a la definición de la ontología como una rica descripción de:

- términos conceptos y nomenclaturas
- relaciones entre conceptos e individuos
- sentencias distinguiendo conceptos y redefinición de relaciones

Según (Hoffman, 2020) se propone la representación del espectro del conocimiento y las capacidades de razonamiento.

Figura 2.1 Espectro para la representación de conocimiento y la capacidad de razonamiento.



Fuente: Hoffman Charles (2020).
Nota: Esta figura detalla las formas de representación del conocimiento para su respectivo tratamiento, en función a la dimensión: Sistema Lógico y la dimensión Capacidad de Razonamiento.

Tal como se puede observar en la figura 2.1, a través de procesos ontológicos, la correlación de las capacidades de un sistema lógico para representar el conocimiento de forma legible para ser interpretado adecuadamente por una máquina y respectivamente y/o como consecuencia la capacidad de que software especializado interprete dicha representación.

Finamente, a este respecto, se concluye que epistemológicamente, la investigación resulta relevante, en vista que tanto los sistemas de Inteligencia Artificial y los esquemas de representación de conocimiento tradicional-formal se sustentan en los mismos principios o relaciones según corresponda.

2.2. Antecedentes de investigación

Se describirá investigaciones relacionadas con modelos soportados por técnicas de Machine Learning para investigaciones relacionadas al Procesamiento de Lenguaje Natural, así como investigaciones en las cuales se basan dichos modelos actuales. Luego, se describirán estudios relacionados con los métodos de pre procesamiento y procesamiento.

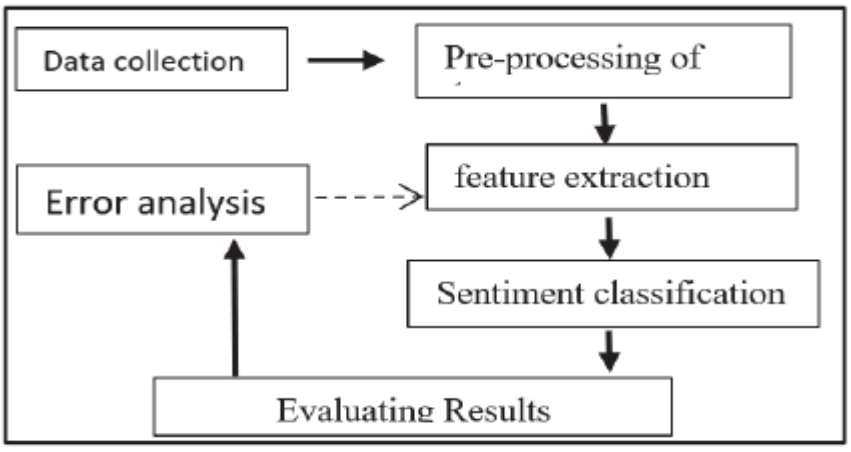
De acuerdo con Subramanian et al. (2021), proponen una investigación relativa al análisis léxico de un dataset relativo a comentarios sobre películas. En la investigación se hace uso de diversas métricas como exactitud, precisión, recall, f1-score.

Como parte del marco metodológico, el flujo de actividades que llevan a cabo los investigadores son:

a. Recolección de data
b. Pre procesamiento de los datos
c. Extracción de características y selección
d. Clasificación de sentimiento
e. Evaluación de resultados y análisis de errores

En la figura 2.2 se destaca el flujo de análisis de sentimiento en detalle.

Figura 2.2 Flujo de análisis de sentimiento.



Fuente: Subramanian et al. (2021)
Nota: La figura detalla, de forma genérica, el proceso de tratamiento de data textual, desde el proceso de recolección hasta la evaluación de resultados.

Los investigadores destacan instrumentos como el enfoque de diccionario léxico, el enfoque de Machine Learning y una combinación de estos dos últimos instrumentos. Como parte del pre procesamiento implica el tratamiento de un dataset alimentado por diferentes usuarios con perspectivas diferentes, esto implica la eliminación de URL, símbolos especiales, emoticones y hashtags. Luego del pre procesamiento es preciso implementar la tokenización, que consiste en separar un fragmento de texto en unidades más pequeñas llamadas tokens. Los tokens pueden ser palabras, caracteres o subpalabras lo que permite la clasificación de hasta 3 tipos: tokenización de palabra, carácter y subpalabra (caracteres n-gram). Como parte del marco metodológico, se elabora una matriz denominada term-document; con esta matriz se puede calcular la similitud de documentos, mediante una multiplicación de matrices Matriz S = DDT lo que permitirá obtener una medida (no normalizada) de similitud. Finalmente, a este respecto, los autores toman en consideración los algoritmos de Inverse Document Frecuency (IDF), análisis Naive Bayes, Support Vector Machine, Sequence-Based NN Clasifiers, Recurrent Neural Networks y Convolutional Neural Networks.

Como parte de las conclusiones, realizan una evaluación de performance entre todos los algoritmos mencionados en el párrafo anterior, llegando a determinar que el algoritmo clasificador de redes neuronales convulocionales (Neural Networks y Convolutional Neural Networks) tiene alta performance, respecto a otros algoritmos para el caso tratado.

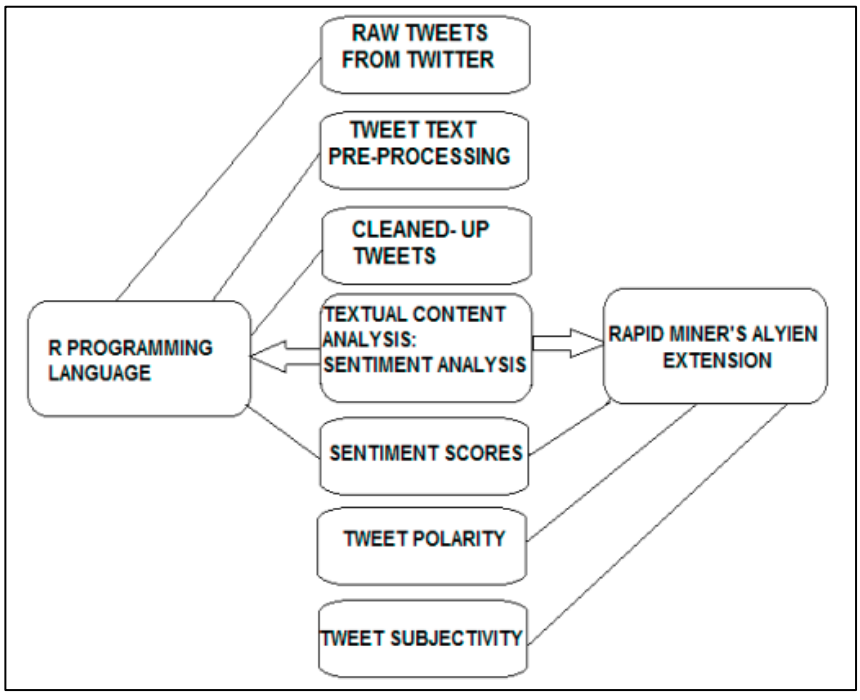
Los investigadores (Sharma y Ghose, 2020), proponen una investigación en la que hacen análisis de sentimiento mediante procesos de scraping del microblogin Twitter en el periodo electoral comprendido entre enero 2019 y marzo 2019. El análisis de sentimiento implicó la obtención de la polaridad de opinión de la gente con respecto a las elecciones. Se consideraron dos candidatos con alta perspectiva de ganar las elecciones.

Como parte del marco metodológico, la primera etapa consistió en la comprensión de los datos (revisión de los datos), esto incluye la recopilación de datos, la descripción y exploración de los datos y verificación de su calidad. Dado que la adquisición de los datos se realiza a través de una fuente en línea (red social), implica la eliminación del ruido para fines de análisis, por lo que se requiere su procesamiento previo. La limpieza implica la

conversión de los feeds de Twitter en minúsculas, la eliminación de las palabras vacías en inglés, URL, números, signos de puntuación y caracteres alfanuméricos. Después del proceso de limpieza el dataset se convirtió en corpus para ser tratado a través de un archivo CSV. La segunda etapa, consistió en el modelado de los datos, lo que incluye la elección de la técnica pertinente. Las entidades nombradas se extraen de los tweets porque con la ayuda de este se puede acceder a cualquier situación dando información en profundidad sobre los nombres de personas, ubicaciones y organizaciones en los tweets, lo que permite a otros revisar rápidamente los tweets y descartar los que son irrelevantes e innecesarios.

Finalmente, como última etapa, la evaluación de todos los tweets no es factible, por lo tanto, se extraen las entidades que son útiles, el análisis se hace sobre la polaridad del sentimiento, la fuerza del sentimiento y la subjetividad del sentimiento, para esto se utiliza AYLIEN, que es una extensión de análisis de texto de Rapid Miner. En la figura 2.3 se describe de forma amplia el proceso metodológico.

Figura 2.3 Metodología general de tratamiento de data de una red social



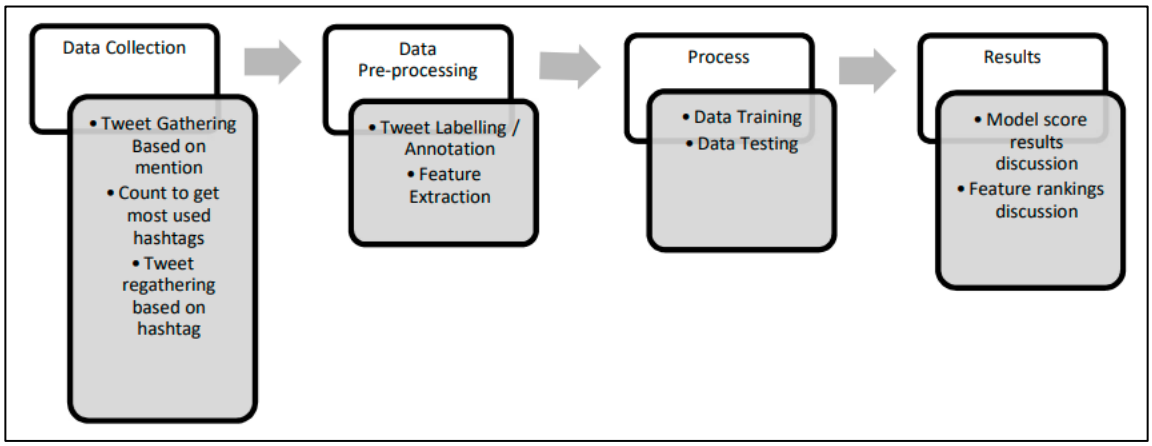
Fuente: Sharma y Ghose (2020)
Nota: Esta figura detalla, la metodología usada por lo autores para el tratamiento de data textual, que tiene como fuente la red social Twitter, con el propósito de hacer análisis de sentimiento.

Como parte de las conclusiones, se propone una metodología que considera el procesamiento de lenguaje natural aplicado sobre textos, para el tratamiento de los comentarios. Se llega a determinar la importancia del grado de popularidad de uno de los candidatos en el uso de la red social, lo que incide sobre el proceso de elección.

Los investigdores (Gumelar y Aini, 2019), proponen una investigación en el que se resalta el uso de bots sociales para la generación de comentarios automáticos para cambiar el tono de una discusión en el proceso de elección presidencial Indonesio del 2019. Se destaca, adicionalmente el problema en redes sociales relacionadas con la promoción de noticias falsas o campañas negras. CNN Indonesia detectó casos en 2018 en Twitter sobre noticias falsas, en la que se identifica a un usuario de Twitter quien afirmó que uno de los candidatos a gobernador de Java Oriental utilizó este mecanismo para influir en la intención de votos de los ciudadanos. Por lo expuesto en este punto, el problema central es como los bots sociales pueden afectar la opinión de los usuarios sobre un tema tan relevante como la elección de una autoridad gubernamental.

Como parte del marco metodológico, se proponen cuatro pasos para el proceso de análisis. El primer paso es la recopilación de datos para su procesamiento (tweets). El segundo paso es el pre procesamiento de datos, que consiste en etiquetar la cuenta manualmente para decidir si una cuenta de Twitter es un bot o no, y extraer las características. El tercer paso es el procesamiento de datos, lo que implica el entrenamiento del conjunto de datos y ejecución de algoritmos de Machine Learning. Como cuarto paso se hace análisis de optimalidad del puntaje basado en el resultado del algoritmo que se produjo a partir del procesamiento de datos. En la figura 2.4 se describe la metodología en general.

Figura 2.4 Metodología para la detección de un social bot.



Fuente: Gumelar y Aini (2019)
Nota: Esta figura detalla el procedimiento regular para el proceso de tratamiento de datos textuales, en este caso la fuente es Twitter en el contexto de la contienda electoral celebrada el 2019 en Indonecia

Como parte de las conclusiones, se afirma que los mejores clasificadores son los algoritmos Suppport Vector Machine (SVM) y Random Forest, que típicamente se usan para procesos de aprendizaje supervisado. Estos dos modelos se entrenaron mediante validación cruzada para mejorar la puntuación general. De estos dos modelos, Random Forest tiene una puntuación general más alta que SVM.

Los investigadores (Bodaghy y Oliveira, 2020), realizan una investigación sobre los difusores de rumores para el cambio de opinión. Los rumores en redes sociales son parte del día a día cuyos impactos no se pueden descartar. Los rumores en las redes sociales se pueden identificar con un conjunto de atributos que es necesario aclarar antes de hacer cualquier análisis en esta línea. Se detectan dos componentes en la investigación el rumor y el antirumor, los cuales se difunden simultáneamente en la red.

Como parte del marco metodológico, un primer componente que consideran relevante los autores es el análisis de datasets sobre rumores y sobre los alcances en función a contextos específicos. Un segundo componente es el análisis de las características de los mensajes, para esto, esto permite evidenciar que las métricas de viralidad funcionan como señales persuasivas que influyen en las percepciones de los usuarios de los mensajes de los medios y sus comportamientos. En el contexto de Twitter,

las métricas de viralidad que acompañan a los tweets sobre un rumor pueden tener un impacto significativo en la confiabilidad percibida por los usuarios de un mensaje dado y en las intenciones de compartirlo con otras personas. El tercer componente en la investigación es el relativo al análisis de difusor y/o su categorización. Los difusores de rumores se comparan como los no difusores de rumores (difusores de noticias verificados) que conduce al estudio sobre la credibilidad del usuario. El análisis de credibilidad utiliza información sobre la cantidad de seguidores que se cree que es una buena característica para distinguir las cuentas maliciosas de otros usuarios, ya que tienen muchos más seguidores. El cuarto y último componente es la detección de macro patrones, que consiste en la identificación de conspiradores de propagación. El investigador hace un análisis diferencial con respecto al difusor. Este análisis se realiza mediante el análisis de centroides.

Como parte de las conclusiones, el proyecto posibilitó la identificación de hasta 3 tipos de usuarios, el usuario difusor, los seguidores y finalmente los usuarios que se dan el trabajo de contrarrestar el rumor. El objetivo de esta investigación fue la evaluación del comportamiento a través de la metodología: Teoría del Comportamiento Planeado (TPB). Como resultado, a través de la metodología propuesta, llega a la identificación de los 3 tipos de usuarios en especial los difusores.

El investigador (Bravo, 2017), realiza una investigación doctoral, relativa al proceso de exploración y explotación de comentarios en Twitter para procesos de análisis sentimental para procesos de adquisición de conocimiento. El problema sobre el que se trabaja en esta investigación es la evaluación de la polaridad de mensajes en tweeter para efectos del proceso de clasificación en categorías de opinión. Se tiene como característica que la técnica de aprendizaje es del tipo supervisado, dado que el corpus de ejemplo tomó como referencia un vector de características elaborados a “mano”

Como parte del marco metodológico, Bravo hace uso de Recursos Léxicos para análisis de sentimiento dentro de los cuales se destaca OpinnionFinder o MPQA Subjectivity Lexicon. Este es un sistema que detecta automáticamente oraciones subjetivas en corpus de documentos. Un grupo de anotadores humanos etiqueta palabras y frases de un corpus de documentos y de acuerdo con las clases de polaridad positiva,

negativa y neutra, esta es la razón por la que se afirma que el proceso de aprendizaje es supervisado. Seguidamente hace uso de ANEW LexiconTheAffective Norms for English Wordslexicon. Esta referencia léxica proporciona ratios emocionales para alrededor de 1.000 palabras en inglés. Estas calificaciones se calculan de acuerdo con tres reacciones psicológicas diferentes de una persona a una palabra específica: valencia (el nivel de agrado), dominio (el grado de control) y excitación (la intensidad de la emoción). La reacción "valencia", que varía en la escala de agradable a desagradable, es el valor más útil para el cálculo de la polaridad. Adicionalmente toma como referencia AFINN Lexicon, SentiWordNet Lexicon, Harvard General Inquirer, NRC word-emotion association Lexicon, NRC-Hashtag. Bravo hace una evaluación de pertinencia en términos de clasificación para la propuesta de su framework, considerando inclusive el análisis temporal de opiniones vertidas fundamentalmente en redes sociales.

Como parte de las conclusiones, Bravo atiende el problema de clasificar los tweets en clases de sentimiento cuando las etiquetas son escasas, para esto desarrolla modelos de inducción de léxico de polaridad y supervisión a distancia. La contribución principal se centra en el modelo de asociación palabra-sentimiento y el modelo de centroide de tweets, esto para la adquisición de conocimiento léxico mediante la inducción automática de léxicos de opinión específicos de Twitter.

El investigador (Haddi, 2015), realiza una investigación doctoral, relativa al análisis sentimental, mediante el uso de técnicas estadísticas y Machine Learning, pero parte del hecho de que el proceso de análisis sentimental fundamentalmente es un problema de clasificación, destaca el hecho de que el alcance de la investigación puede centrarse en diferentes frentes como el marketing y generación de nuevas fuentes de conocimiento.

Como parte del marco metodológico, Haddi hace una propuesta de un framework que consta de 3 etapas, en la primera etapa se parte por la extracción de las características más relevantes empleando procesos de transformación y filtrado exhaustivos. En segundo lugar, se propone una matriz de características y se realiza un cálculo de precisiones. Finalmente se realiza una evaluación del rendimiento del clasificador. Se observa el uso de los métodos Feature Presence (FP), Feature Frecuency (FF) y Term Frecuency –

Inverse Document Frecuency (TF-IDF) esto, para la selección de características. Para el proceso de clasificación hace uso de la técnica de Support Vector Machine (SVM).

Como conclusión, mediante el proceso de técnicas de aprendizaje automático, para escenarios como el de revisiones de comentarios dispuestos en diversos medios sobre películas, hasta las revisiones sobre las publicaciones en diarios sobre el comportamiento financiero, los resultados son prometedores, con posibilidad de alcance al marketing, respuesta automática de preguntas (bots) y base de conocimiento. En la investigación hace uso del procesamiento de lenguaje natural para el tratamiento de datos. Como conclusión, dadas las características computacionales de la época, los resultaron ser prometedores en especial para el desarrollo aplicaciones vinculadas al marketing.

Los investigadores (Di Cocco y Monechi, 2021) proponen un método de análisis textual mediante técnicas supervisadas de Machine Learning, para la determinación de cuan populista puede ser una determinada propuesta política. Como parte del método se utiliza un algoritmo de clasificación capaz de discriminar entre diversas frases textuales de partidos populistas o no. El algoritmo clasificador fue RandomForest para efectos de asegurar predicciones precisas para relaciones no lineales.

Como conclusión en esta investigación, a diferencia de procedimientos de análisis textual asistido por computadora (Clasificación holística de Hawkins) el proceso se basa en la observación de unidades de texto (vectorizando párrafos). En la investigación se llega a medir los niveles de populismo de los partidos y no de los líderes. Otro aspecto importante de esta investigación es que para la clasificación no se basa en diccionarios con lo que se reduce el riesgo de clasificaciones arbitrarias.

2.3. Bases teóricas

2.3.1. Inteligencia artificial

Hablar de los fundadores de la inteligencia Artificial podría ser controversial debido a que, en diferentes momentos de la historia, hubo grandes pensadores que sentaron sus bases. El primero de ellos fue Alan Turing, quien en 1936 ideo un modelo matemático

que reduce la estructura lógica de procesamiento computacional a un estado mínimo que a largo plazo sentó las bases para el procesamiento computacional moderno. Junto con la estructura de la máquina, concibió las bases de la programación lineal y/o secuencial de instrucciones las cuales eran dispuestas en una cinta con capacidad de ser interpretadas y modificadas a través de un cabezal y una unidad central de procesamiento. Con la estructura propuesta, la máquina de Turing se convirtió en la base de todas las computadoras digitales posteriores, que comparten el esquema básico de la máquina de un dispositivo de entrada, un dispositivo de salida (cinta y lector), memoria (para almacenamiento) y unidad central de procesamiento.

Teniendo ya computadoras literales para 1956, John McCarthy acuñaría el término de "Inteligencia Artificial" por primera vez cuando organizó junto con Marvin Minsky, Nat Rochester y Claud Shanon una conferencia sobre este tema, financiado por la Fundación Rockefelle. El objetivo de McCarthy era una primera iniciativa para formar un grupo investigador en el breve plazo posible. El grupo propuso estudiar la relación del lenguaje con inteligencia durante los dos meses del proyecto de 10 personas.

Feigenbaum define la inteligencia artificial como: "componente de las ciencias de la computación que se ocupa del diseño de sistemas informáticos inteligentes, es decir, sistemas que exhiben las características que asociamos con la inteligencia en el comportamiento humano, como comprender el lenguaje, aprender, razonar, resolver problemas, etc."

Según (Russell y Norving, 2008), nos presentan 4 definiciones de IA organizadas en cuatro categorías mostradas en la tabla 2.1:

Tabla 2.1: **Definiciones de IA organizadas en cuatro categorías.**

Sistemas que piensan como humanos	Sistemas que piensan racionalmente
"El nuevo y excitante esfuerzo de hacer que los computadores piensen… máquinas con mentes, en el más amplio sentido literal". (Haugeland, 1985)	"El estudio de las facultades mentales mediante el uso de modelos computacionales". (Charniak y McDermott, 1985)

"La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…" (Bellman, 1978)	"El estudio de los cálculos que hacen posible percibir, razonar y actuar". (Winston, 1992)
Sistemas que actúan como humanos	Sistemas que actúan racionalmente
"El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia". (Kurzweil, 1990)	"El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia". (Kurzweil, 1990)
"El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor". (Rich y Knight, 1991)	"El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor". (Rich y Knight, 1991)

Fuente: Russel y Norving (2008).
Nota: El propósito de lo descrito en la tabla es analizar, mediante un proceso de contrastación basado en definiciones formales, la manera en que los seres humanos piensan y razonan, y cómo se espera que las máquinas reproduzcan estos mismos procesos.

Russel y Norving, ponen sobre relieve la necesidad de configuración del agente racional, equivalente al ente humano que está lleno de sensores y actuadores y sobre todo de una unidad central de procesamiento con capacidad cognitiva y sobre todo con capacidad, en función a un proceso de entrenamiento de hacer inferencia. Este agente racional (no humano) debería reunir precisamente las características del ser humano hasta el grado de hacer inferencia de tal forma que tenga capacidad de hacer elecciones "racionales" frente a escenarios de incertidumbre.

Al margen de lo mencionado hasta este punto, queda claro que hubo diferentes formas de contribución para la configuración de lo que es actualmente la Inteligencia Artificial, las cuales se destacan a continuación:

- Los filósofos (400 AC) concibieron al ser humano como una máquina, que presente determinadas acciones en función a conocimiento. Fueron los filósofos quienes a través de la proposición de silogismos sentaron las bases de la lógica proposicional que luego fue formalizado por los matemáticos.
- Los matemáticos diseñaron los artefactos para el tratamiento de las aseveraciones lógicas y definieron reglas específicas, procedimientos e interpretación; este proceso de formalización posibilitó una evaluación probabilista y adicionalmente el diseño de algoritmos para manipulación.
- Los psicólogos contribuyeron en la IA al entender cómo es que el cerebro estuvo configurado y los neurólogos al identificar la unidad mínima de procesamiento: la neurona, que luego permitió el diseño de la neurona artificial. Fueron los psicólogos quienes asumieron que podría considerarse el cerebro tanto de humanos y animales como máquinas de procesamiento frente a determinados estímulos.
- Los informáticos, contribuyeron en la IA al construir artefactos en términos de hardware y software para desplegar la arquitectura funcional de un sistema inteligente.
- Los lingüistas, contribuyeron en la IA debido a que era preciso tener un esquema de representación del conocimiento. El esfuerzo data desde la aparición de los Sumerios quienes idearon los primeros vestigios de escritura. Con el transcurrir del tiempo se consolida la gramática y la sintaxis. Las reglas gramaticales y sintácticas al estar plenamente maduras, han servido para lo que a la fecha se denomina: Procesamiento de Lenguaje Natural.

Según Xu, X. et al. (2021) destacan la capacidad de la IA en el proceso de imitación mediante computadoras. La inteligencia artificial es un enfoque para edificar máquinas capaces de hacer labores como las realizadas por humanos. Por supuesto, esta es una definición bastante extensa, y puede referirse desde Apple Siri a Google AlphaGo y tecnologías bastante poderosas todavía por crear. Al simular la sabiduría humana, los sistemas de inteligencia artificial acostumbran a mostrar por lo menos algunos de los siguientes comportamientos asociados con la sabiduría humana: planificación,

aprendizaje, argumento, percepción, entre otros y, en menor medida, sabiduría social y creatividad.

2.3.2. Machine learning

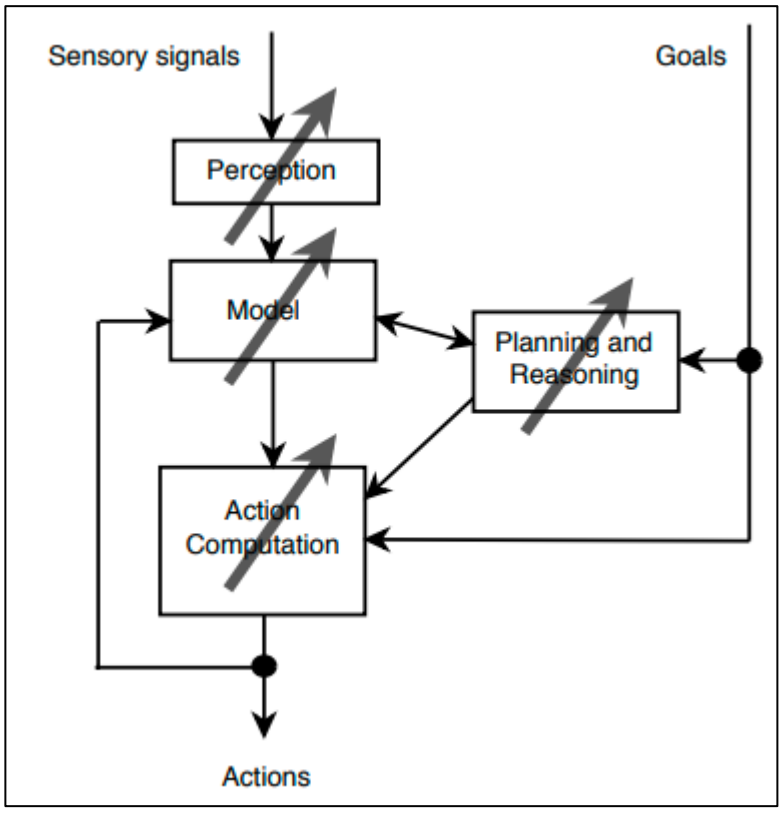
En 1950 Alan Turing propuso el denominado Test de Turing, el cual establece un criterio básico para determinar si un sistema computacional inteligente es detectable o no por una persona. En este escenario, una primera persona mantiene una conversación con una computadora y una segunda persona, pero sin saber quién de los dos conversadores es realmente una máquina; si ninguna de las personas se da cuenta de quién es la computadora, entonces se habrá superado el Test de Turing, lo que permite concluir que la computadora no solo tiene la capacidad de aprendizaje, sino que tiene la capacidad de interacción emulando todo el comportamiento del ser humano. Se tuvo que esperar hasta el 2012 para que el Test de Turing pasara una prueba exitosa (sin dejar de ser controversial). “Eugene Goostman”, un bot conversacional desarrollado para simular la personalidad de un joven adolescente ucraniano. Esta característica fue crucial durante el test al no imitar la conversación de un adulto, simulaba de forma natural el desconocimiento propio de su edad. Eugene aprobó con un ajustado 33%, que eran los parámetros oficiales establecidos por Turing. Este 33% nos permite inferir que probablemente hubo un 66% de potenciales errores a la hora de enfrentar la prueba.

Una de las expectativas dentro de la IA es que el sistema computacional tenga la capacidad de aprendizaje. En 1957 Frank Rosenblatt diseñaría el algoritmo básico que representaría al perceptron, el cual, es un algoritmo supervisado para procesos de clasificación. Con este algoritmo se da inicio a las redes neuronales. En 1967 se desarrolla el algoritmo “Nearest Neighbor”, con esto se marca un hito en lo concerniente a la detección o reconocimiento de patrones, esto es muy importante debido a que el volumen de datos, gracias a la capacidad de almacenamiento hacía inviable el uso de algoritmos de búsqueda secuencial y se precisaba nuevas estrategias de búsqueda y como consecuencia nuevas estrategias de aprendizaje en el ámbito de la IA.

De acuerdo con (Nilsson, N., 2005) el Machine Learning o también conocido como Aprendizaje Automático, se refiere a los procesos de cambios en un sistema inteligente o

agente inteligente. Los procesos de cambio pueden ir desde reconocimiento, diagnóstico, planificación, predicción, etc. Los "cambios" pueden ser mejoras de sistemas que ya funcionan o síntesis de nuevos sistemas. La arquitectura de un agente inteligente se muestra en la figura 2.5.

Figura 2.5 Representación de un agente en IA



Fuente: Nilsson, N. (1998)
Nota: La imagen representa un modelo general de arquitectura de un agente inteligente, muy usado en Inteligencia Artificial y robótica. Muestra cómo un sistema recibe información del entorno, la procesa, decide qué hacer y ejecuta acciones para alcanzar ciertos objetivos.

Este agente percibe y modela su entorno a través de los sensores que forman parte de su estructura y computa las acciones apropiadas, de acuerdo al contexto cambiante procurando anticipar sus efectos. Los cambios realizados en cualquiera de los componentes pueden dar a entenderse como proceso de aprendizaje. Se pueden emplear diferentes mecanismos de aprendizaje según el subsistema que se cambie.

A continuación, se detallarán las principales fuentes sobre las que se sustenta el aprendizaje automático.

- **Las estadísticas:** Una estrategia conocida para procesos de inferencia, es la toma de muestras de una población grande a partir de distribuciones de probabilidad desconocidas. Trasladando esta perspectiva, el problema es estimar el valor de una función desconocida en un nuevo punto dados los valores de esta función en un conjunto de puntos de muestra. Como se puede observar, los métodos estadísticos para estos tipos de problemas, donde los puntos de datos son demasiado grandes pueden ser considerados como instancias de aprendizaje automático debido a que las reglas de decisión dependen del tratamiento de las muestras.
- **Modelos cerebrales:** Como se mencionó en el punto 2.3.1 los psicólogos y puntualmente los neurocientíficos, teniendo como objetivo el entendimiento de la funcionalidad del cerebro como máquina, redujeron su análisis al identificar al componente mínimo elemental: la neurona. Una vez identificada la neurona como unidad básica de procesamiento, (McCulloch y Pitts, 1943), simularon el comportamiento de una neurona biológica asignando pesos aleatorios a sus entradas para configurar un modelo lineal. Teniendo la configuración de varias neuronas como grupo de trabajo, se gesta uno de los esquemas sobre los que se sustenta los algoritmos de aprendizaje automático: las redes nuronales. Ocasionalmente este esquema o estructura de análisis basado en redes neuronales se denomina conexionismo, computación al estilo del cerebro o procesamiento subsimbólico.
- **Teoría de control:** Los sistemas biológicos interactúan en un contexto completamente cambiante, para esto, apelan a innumerables sensores que les permite aprender y reaccionar para garantizar su supervivencia. Mientras más interacción se tenga con el entorno o contexto se tendrá mayor capacidad de aprendizaje. El proceso de adaptación, propio de los sistemas biológicos, se sustenta a través de mecanismos de control. Los encargados de estudiar los mecanismos de control estudian el problema de controlar un proceso que tiene parámetros desconocidos los cuales deben estimarse que ocasionalmente cambian durante el funcionamiento dado que los contextos o entornos son cambiantes. Es

en la robótica donde se observa el uso de los mecanismos de control a través de entradas sensoriales.

- **Modelos psicológicos:** Son los psicólogos quienes mientras estudiaban la forma en que se aprende (no necesariamente seres humanos) posibilitó la definición de los árboles de decisión, inclusive el aprendizaje por refuerzo se atribuye al esfuerzo de los psicólogos cuando utilizaron mecanismos de recompensa para el estímulo del aprendizaje.
- **Modelos evolutivos:** La capacidad de adaptación en la naturaleza se da gracias a la capacidad de evolución aunque esto no tenga nada que ver con las habilidades cognitivas que pueda tener el agente. No se tiene un modelo que permita representar la distancia entre evolución y aprendizaje, pero a través de los biólogos, se tuvo la comprensión de la capacidad evolutiva de los agentes, esto a través del entendimiento de la fusión de cromosomas. Una vez entendidos estos principios, Holland en 1975 diseña lo que se denomina los algoritmos genéticos y la programación genética que a la fecha resultan ser las técnicas computaciones más robustas para procesos de aprendizaje.

Según (Russell y Norving, 2008) manifiestan que para el proceso de aprendizaje se han creado una gran variedad de elementos de soporte que se sustentan en:

- Qué componente del elemento de acción tiene que aprenderse
- Qué realimentación está disponible para aprender dicho componente
- Qué tipo de representación se usa para el componente

Una de las características de los sistemas abiertos es la capacidad de retroalimentación para el proceso de aprendizaje por lo que se concluye que el tipo de retroalimentación afectará directamente el aprendizaje, a este respecto se tienen identificados 3 tipos de aprendizaje:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje por refuerzo

2.3.2.1. Aprendizaje supervisado

El aprendizaje supervisado recibe dicha denominación debido a la configuración de una función encargada de entregar valores de entrada a otra función para efectos de entrenamiento, precisamente este es el momento en que el agente aprende. El proceso de entrenamiento se da con el objetivo de que el agente sepa devolver un conjunto de valores (denominado señal de supervisión). Desde de la perspectiva de la automatización, el algoritmo de aprendizaje supervisado analiza los datos de entrenamiento y produce la salida de inferencia deseada, que puede usarse para analizar nuevas entradas, debido a esto último es que un bloque de datos se divide en datos de entrenamiento y datos de prueba. El algoritmo de aprendizaje debe tener la capacidad de generalizar a partir de datos de entrenamiento a situaciones no evaluadas de manera "razonable" (sesgo inductivo). El proceso de entrenamiento no implica necesariamente la obtención de un resultado infalible, esto significa que debe tenerse un margen de error estadísticamente aceptable.

Un algoritmo genérico para la aplicación de aprendizaje supervisado es el siguiente:

a. De la población de datos disponible debe haber un proceso de selección muestral estadísticamente razonable, se asume que estos datos muestrales deben ser representativos.
b. El proceso de entrenamiento se dará explícitamente asignando un conjunto de salidas provistas por una función a las entradas del agente que al mismo tiempo devolverá un resultado que se ajusta a la realidad.
c. Evaluar la pertinencia de lo asignado y el resultado del agente, con esto se busca incrementar la precisión.
d. Determinar la estructura de la función de aprendizaje aprendida y el algoritmo de aprendizaje.
e. Dependiendo del algoritmo específico de aprendizaje será necesario entrenar este último todas las veces que sean necesarias de acuerdo a los parámetros de control y optimalidad.
f. Evaluar la precisión

El aprendizaje supervisado puede dividirse para dos tipos de problemas:

- Clasificación
- Regresión

Los principales algoritmos para aprendizaje supervisado se describen a continuación.

- Support Vector Machine (SVM)
- Regresión logística
- Random Forest
- Naive Bayes Classifier
- K-Nearest-Neighbor (KNN)

2.3.2.2. Aprendizaje no supervisado

El aprendizaje no supervisado, a diferencia del aprendizaje supervisado no aprende a partir de la asignación de etiquetas. Este tipo de aprendizaje tiene como principal característica auto organizar dentro del bloque de datos a partir de la identificación de patrones utilizando métodos probabilísticos o combinaciones de preferencias. La capacidad de descubrir similitudes y diferencias en la información (patrones) lo convierte en una solución conveniente para análisis exploratorio, cruce de datos, segmentación y atención a problemáticas relativas al reconocimiento de imágenes (computación gráfica)

Las principales características de este método de aprendizaje son:

- Encuentra todo tipo de patrones desconocidos en los datos.
- Ayuda a encontrar características que pueden ser útiles para la categorización.
- El análisis puede ser realizado en tiempo real por lo que todos los datos finalmente se etiquetarán
- La intervención humana es secundaria, debido a que es más fácil obtener datos sin etiquetar en el proceso de análisis algorítmico.

A continuación, se detallan algunas aplicaciones del aprendizaje no supervisado.

- Clustering
- Detección de anomalías

- Asociación minera
- Modelos de variables latentes
- Análisis de componente principal

2.3.2.3. Aprendizaje con refuerzo

Este método tiene como característica recompensar al agente inteligente en la medida que se cumplan con determinados objetivos y de la misma forma aplica un esquema de penalidad frente al incumplimiento de objetivos. Este esquema, estimula al agente para que busque una la máxima recompensa y con ello alcanzar la solución óptima.

Dentro del ambiente del proceso de aprendizaje, este esquema, estimula a que el agente se concentre en las soluciones óptimas.

A continuación se describirán los principales instrumentos para la implementación de esta forma de aprendizaje.

- Basado en valores: Esta función procura encontrar puntos óptimos que posteriormente serán usados para su evaluación. Con esto el agente se concentra solo en las variables latentes.
- Basado en políticas: Esta función (a diferencia de la anterior) procura encontrar el procedimiento o política óptima para obtener las máximas recompensas futuras. Con este enfoque, en la ejecución del algoritmo, los procedimiento o políticas de recompensa pueden ajustarse dinámicamente. Existen dos enfoques para el diseño de la política:

  Determinista: Produce la misma acción en cualquier estado.

  Estocástico: Define su aplicabilidad en función a la probabilidad de la acción producida.
- Basado en modelos: Se genera un modelo o entorno virtual y es el agente que explora dicho entorno para aprenderlo. El diseño del modelo virtual, dependerá del contexto, por lo que no existe un procedimiento específico para el diseño universal del entorno virtual.

Se presenta dos tipos de algoritmos de refuerzo:

a. Aprendizaje por refuerzo positivo

El aprendizaje por refuerzo positivo es una estrategia algorítmica a través de la cual se identifican eventos favorables y se establece un sistema de recompensa en caso se de este escenario. El agente aumenta la fuerza y la frecuencia del comportamiento y tiene directamente un impacto positivo. Este método algorítmico maximiza el rendimiento y mantiene el cambio durante un período prolongado, es preciso resaltar que es necesario monitorear y/o evaluar el proceso debido a que demasiado refuerzo ocasionaría optimización excesiva.

b. Aprendizaje por refuerzo negativo

El aprendizaje por refuerzo negativo es una estrategia algorítmica a través de la cual se identifican eventos no favorables y se establece un sistema de recompensa en caso se de este escenario para evitar que se repitan. A diferencia del aprendizaje por refuerzo positivo que busca la optimización el de refuerzo negativo busca determinar el estándar mínimo de rendimiento.

Algunos de los beneficios de esta técnica de aprendizaje son:

- Tiene un comportamiento holista
- No requiere un proceso de recopilación de datos por separado
- Aplicable a entornos dinámicos con incertidumbre

2.3.3. Procesamiento de leguaje natural

El procesamiento del lenguaje natural, en inglés Natural Language Processing (NLP) se refiere al método usado por parte de los sistemas basados en la inteligencia artificial para comunicarse a través del lenguaje natural que usan los seres humanos.

El NLP desplaza a los primitivos esquemas de uso de comandos y procura, sobre todo para sistemas robóticos, recibir las instrucciones no a base de comandos sino por instrucciones verbales que resultan naturales al ser humano. Se espera que los sistemas NLP tengan la capacidad de leer y escribir.

Un elemento relevante en NLP es el CORPUS. El corpus es la colección representativa y estructurado de textos que se han producido en un entorno de comunicación natural. La fuente del corpus puede ser texto electrónico, transcripciones y reconocimiento óptico de caracteres e inclusive transcripciones orales en línea. El corpus se utiliza únicamente para el análisis lingüístico estadístico y en función a la naturaleza de la investigación para prueba de hipótesis.

Los componentes de un sistema NLP son:

Entendimiento del lenguaje natural:

- Analizar la entrada, que se asume es en lenguaje natural, estos, deben cumplir con el requisito de ser lo mínimamente entendibles (aunque esto es relativo, sobre todo cuando la fuente de datos por ejemplo son las redes sociales)
- Análisis estructural del lenguaje.

Generación del lenguaje natural:

- Analiza la salida, que se asume debe ser diseñada para alguien que hace uso del lenguaje natural. Puede implicar:
    - La planificación del texto: Toma como referencia una base de conocimientos para el proceso de construcción del texto.
    - La planificación de oraciones: Define el conjunto de palabras clave para la construcción de la expresión.
    - Realización de texto: Define la estrategia de construcción de la oración en función a los dos pasos previos.

Los pasos para el análisis de lenguaje natural son (ver figura 2.10):

a. Análisis léxico: Identifica la estructura del uso de las palabras, esto puede variar en función a la estructura lexica del lenguaje en cuestión. En el análisis léxico se determinará por fragmentos, párrafos, oraciones y palabras.
b. Análisis sintáctico (También llamado parsing) En función al análisis previo y la estructura gramatical de la lengua o idioma se determina la relación entre las palabras usadas en una determinada sentencia. La estructura gramatical de una

oración posibilitará, a través del sistema basado en PLN descartar oraciones que no cumplan las reglas estructurales.

c. Análisis semántico. Como su nombre lo indica, en esta etapa se realiza un análisis en función al significado para lo cual se apela a un "diccionario semántico". El significado de lo expresado estará supeditado a las reglas y las variaciones de interpretación en función a las relaciones entre las palabras.
d. Integración de discurso. Identifica el contexto general de la expresión, esto en función al resultado del análisis de las oraciones debido a que el significado de una oración depende del significado de la oración anterior.
e. Análisis pragmático. En esta última etapa se realiza un análisis de lo expresado para determinar lo que realmente significa considerando los aspectos regulatorios y semánticos del lenguaje.

2.3.4. Métodos de representación de texto

2.3.4.1. TF-IDF

El concepto de codificar textos se remonta al problema de hacer que las máquinas puedan ejecutar métodos más automatizados de búsqueda de documentos textuales (Information Retrieval), la solución propuesta por (Luhn, 1957) era un sistema de búsqueda de literatura basado en métodos estadísticos y con un alto grado de mecanización. Dicho sistema estaba basado en una serie de pasos, de entre ellos, se destaca la "codificación automática", cuya finalidad era la de crear patrones con los documentos codificados que luego podrían servir para medir el grado de similitud conceptual entre documentos.

Según (Jurafsky & Martin, 2023) mencionan que entre más frecuente sea una palabra o una combinación de palabras dentro de un documento, más alta es la probabilidad de que estas sirvan para representar la idea principal de tal documento. Aunque, por otro lado, también menciona que palabras comunes en un campo puede tener poca o nula relevancia (ej. la palabra "electricidad" apareciendo en documentos especializados en el campo de electricidad).

Del párrafo anterior se desprende la idea inicial de ponderación de términos conocida como term-frequency, que de acuerdo con la definición original propusta por Luhn (1957), es la frecuencia de la palabra t en el documento d y se puede definir mediante la siguiente ecuación:

$$tf_{t,d} = count(t, d)$$

Donde:

t: Una palabra en particular

d: Un documento/texto en particular

La función $count()$ representa al conteo simple de palabras. El conteo simple es el número de veces que una palabra en particular aparece en un documento en particular, esto a su vez, da paso a la matriz *term-documents*. Un ejemplo de esta matriz se encuentra en la tabla 2.2.

Tabla 2.2: Matriz term-documents basada en obras de Shakespeare

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
Battle	1	0	7	13
Good	114	80	62	89
Fool	36	58	1	4
wit	20	15	2	3

Fuente: Adaptado de Jurakksy (2023)
Nota: La tabla muestra cuántas veces aparecen ciertas palabras en cuatro obras de William Shakespeare. Se observa que términos como “battle” aparecen más en obras históricas como Henry V y Julius Caesar, mientras que “fool” y “wit” son más frecuentes en comedias como As You Like It y Twelfth Night. Esto refleja cómo el vocabulario cambia según el tema y género de cada obra.

Cada celda de la tabla 2.2. representa el resultado de la función $count()$ y por ende el resultado de term-frequency. En la tabla se han usado 4 palabras y 4 obras de Shakespeare donde, a modo de ejemplo, la palabra “battle” ($t =$ “battle”) aparece 7 veces

en el documento “Julius Caesar” (d = “Julius Caesar”), por ende, $tf_{battle,Juliuis\ Caesar} = 7$.

Adicionalmente, cada columna de la matriz en la tabla es la representación vectorial del documento, del mismo modo, cada fila es la representación vectorial de cada palabra, cada representación vectorial convive en un espacio vectorial por lo que es posible comparar su similitud (palabras entre palabras o documentos entre documentos) utilizando una métrica de distancia entre vectores como la distancia de coseno que se define como:

$$dcoseno(v, w) = v \cdot w = \cos(\theta)$$

Donde:

v y $w \in \mathbb{R}^n$ son vectores normalizados (unitarios) en el espacio vectorial $\mathbb{R}^n$

n: Dimensión del vector

θ: Ángulo que forman los vectores v y w

El segundo factor en $tf\text{-}idf$ sirve para ponderar con mayor peso a las palabras que ocurren con poca frecuencia en un documento y con menor peso en el caso contrario, esto está basado en que las palabras que aparecen muchas veces en un mismo documento no son tan relevantes. Esta ponderación fue propuesta por Jones (1972) y es conocida como idf (inverse document frequency) la cual se define como N/df_t donde N es el número de documentos totales en una colección de documentos (corpus), y df_t, es el número de documentos en que una palabra t aparece.

Acorde con Luhn (1953), debido a la gran cantidad de documentos que existen en una colección, la ponderación se suele reducir mediante una función logarítmica, de modo que la definición de idf es:

$$idf_t = \log_{10}(\frac{N}{df_t})$$

Donde:

t: Una palabra en particular

N: Número total de documentos en una colección

df_t: document-frequency, número de documentos en los que aparece la palabra t

$tf\text{-}idf$ es el término numérico de ponderación $w_{t,d}$ para la palabra t en el documento d que combina a $tf_{t,d}$ con idf_t por lo tanto, se define como:

$$w_{t,d} = tf_{t,d} \times idf_t$$

Donde:

$tf_{t,d}$: Frecuencia de t en el documento d (term-frequency)

idf_t: Inverse document frequency de t

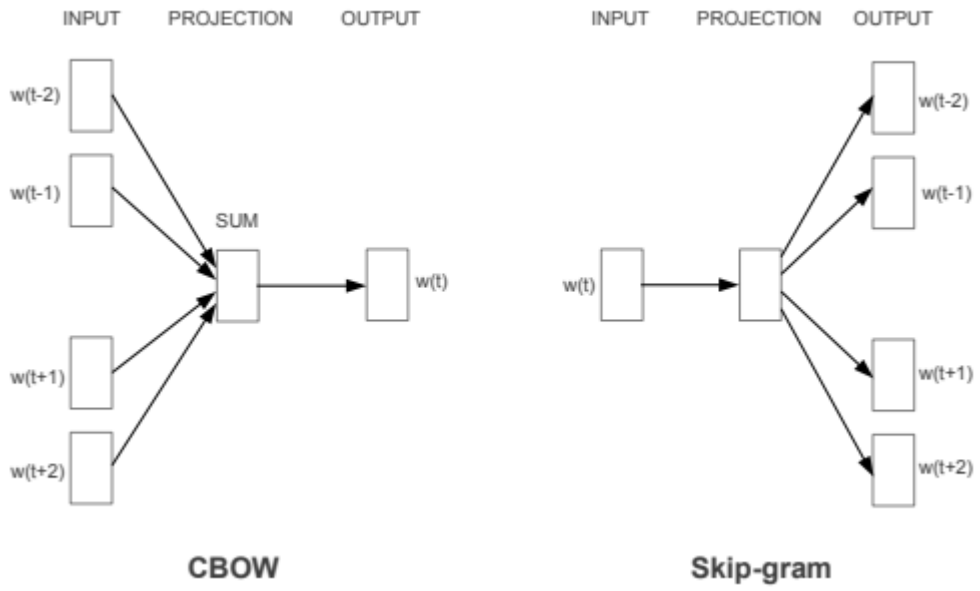
2.3.4.2. Word2Vec

Los enfoques de representación de texto por frecuencia como los mencionados en el punto anterior (tf y $tf\text{-}idf$) tienen una debilidad: representan palabras en vectores de dimensiones que usualmente toman todo el conjunto V de palabras en el corpus de entrenamiento, resultando en vectores dispersos (sparce vectors).

Word2Vec es otro método de representación vectorial que emplea *embeddings* como vectores semánticos de representación. Los embeddings son vectores densos (dense vectors) y de tamaño reducido (usualmente de dimensiones que van desde 50 a 1000), y aunque estas dimensiones no tienen una interpretación clara hasta el momento, Jurafsky & Martin, 2023 afirma que los embeddings funcionan mejor en todas las tareas relacionadas a NLP posiblemente porque los clasificadores tienen que entrenar menos parámetros, lo que reduce el espacio de búsqueda y a su vez, esto aumenta la posibilidad de que los modelos generalicen mejor.

Word2Vec es un conjunto de algoritmos propuestos por Milokov et al. (2013), específicamente, dentro de dicho conjunto existen dos algoritmos capaces de representar vectorialmente en embeddings a un corpus de palabras, estos son: CBOW (Continuous Bag Of Words) y Skip-gram, este último también conocido como SGNS (Skip-gram with negative sampling). La figura 2.6 muestra las arquitecturas de los dos algoritmos presentes en Word2Vec.

Figura 2.6 Arquitectura de CBOW y Skip-gram en Word2Vec.



Fuente: (Mikolov et al., 2013)
Nota: La imagen compara dos modelos de representación de palabras: CBOW y Skip-gram. En CBOW, el modelo usa las palabras del contexto para predecir la palabra central $w(t)$. En Skip-gram, ocurre lo contrario: a partir de la palabra central se predicen las palabras que la rodean en el contexto.

De forma general, CBOW predice la palabra actual $w(t)$ dado como input el contexto cercano a la palabra actual, mientras que Skip-gram predice el contexto cercano de palabras dada la palabra actual, esto sergún Milokov et al. (2013).

Para el caso de skip-gram, Word2Vec entrena a un clasificador en una tarea de predicción binaria que trata de predecir qué tan probable es que una palabra $w(t+1)$ aparezca cerca de $w(t)$ (que esté dentro su contexto), los pesos obtenidos de dicho entrenamiento son los llamados embeddings de la palabra $w(t)$. Además, este entrenamiento se puede realizar sin supervisión, es decir, es de tipo no supervisado, ya que cualquier palabra que esté dentro del contexto de $w(t)$ puede actuar como ground

truth o label verdadera. Esta ventaja hace posible que Word2Vec pueda ser entrenado en corpus de miles de millones de palabras de forma automática.

2.3.4.3. Skip-gram

De acuerdo con Milokov et al. (2013), el objetivo del modelo Skip-gram es el de maximizar la probabilidad logarítmica:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log(p(w_{t+j}|w_t))$$

Donde:

$T =$ Tamaño del corpus

$c =$ Tamaño del contexto (window) de entrenamiento

$w_t =$ Palabra central/fijada, dada la palabra central, se quiere encontrar la probabilidad de que otra palabra w_{t+j} se encuentre dentro del contexto de entrenamiento delimitado por $[-c, c]$.

$j =$ Índice que se añade a la ubicación de la palabra central w_t dentro del contexto de entrenamiento.

Como se mencionó anteriormente, la tarea de Skip-gram es la de predecir que tan probable es que la palabra w_{t+j} esté en el contexto de w_t, formalmente esto está representado como:

$$p(w_{t+j}|w_t)$$

Asimismo, hay varias maneras de definir $p(w_{t+j}|w_t)$, la forma más básica es usando la función softmax (regresión logística multinomial), sin embargo, usar esta representación lleva a un alto grado de complejidad algorítmica. Para esto, los autores

emplearon dos variaciones, una de ellas es la función softmax jerárquica (Hierarchical softmax) (Morin y Bengio, 2005) que emplea un árbol binario para calcular las probabilidades, de modo que la complejidad se vea reducida a una función logarítmica.

La otra técnica consiste es usar Muestreo Negativo (Negative Sampling), una técnica basada en Noise Contrastive Estimation (NCE) (Gutmann y Hyvarinen, 2012) que emplea una regresión logística para diferenciar entre la palabra objetivo y palabras que estén fuera de su contexto (muestras "negativas"), las cuales siguen una distribución $P_n(w)$. De este modo, la probabilidad logarítmica se puede definir como (Jurafsky (2023)):

$$\log p\left(w_{t+j}|w_t\right) = \sum_{i=1}^{L} \log \sigma\left(w_{t+j} \cdot w_t\right)$$

Donde:

L: Tamaño del contexto (window) que rodea a w_t

Y σ es la función sigmoide representativa de la regresión logística que se define como:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Asimismo, se define a la función objetivo del Muestreo Negativo como (Jurafsky & Martin, 2023):

$$p\left(w_{t+j}|w_t\right) = p(c|w) \; = \; \log \sigma\left(c_{pos} \cdot w\right) + \sum_{i=1}^{k} \log \sigma\left(-c_{neg} \cdot w\right)$$

Donde:

w: Palabra objetivo (Palabra "input")

c: Palabra contexto (Palabra "output"), $-c$ representa a una palabra contexto obtenida de la distribución negativa $P_n(w)$.

k: Número de muestras negativas por cada registro.

De forma intuitiva, lo que se quiere lograr es maximizar el producto punto entre la palabra objetivo y su contexto real y minimizar el producto punto entre la palabra objetivo y las k muestras negativas que no están realmente en el contexto de la palabra objetivo.

2.3.4.4. FastText

La limitación de los algoritmos de Word2Vec recae en que depende de un vocabulario previamente definido, por ende, no puede representar palabras desconocidas o fuera de vocabulario (OOV – Out Of Vocabulary). Esto es una gran limitación debido a que el lenguaje escrito es muy variado y usualmente no es posible entrenar al modelo con todas las palabras existentes de un lenguaje dado, siempre cabe la posibilidad de que algunas palabras queden fuera y, por ende, no puedan ser representadas posteriormente llevando a la imposibilidad de utilizar ciertos test sets en la práctica.

Fastext, según Bojanowski et al. (2017), trata con este problema con el uso de sub-palabras o n-grams. Esta técnica está basada en el trabajo de Milokov et al. (2013), solo que esta vez en lugar de considerar la probabilidad de que la palabra contexto (output) aparezca en la vecindad de la palabra objetivo (input), se obtiene la probabilidad agregada ente los n-grams de la palaba objetivo y la palabra contexto de modo que:

$$s(w, c) = \sum_{g \in G_w} z_g v_c$$

Donde:

$z_g =$ Vector que representa al n-gram g que aparecen en la palabra objetivo w

v_c = Vector output tambien conocido como vector contexto.

$g =$ n-gram g basado en la palabra objetivo w

G_w = Conjunto de n-grams que aparecen en w

Es decir, un embedding de Skip-gram z_g (o CBOW) es aprendido por el modelo por cada n-gram en G_w. Finalmente, la representación vectorial final de la palabra w es dada por la suma de todos los embeddings z_g obtenidos de G_w. De esta forma es posible representar palabras OOV mediante la combinación de sus n-grams.

2.3.5. Algoritmos de clasificación

2.3.5.1. Support Vector Machine

Support Vector Machine (SVM) o también llamado Support Vector Network, inicialmente propuesto por (Cortes y Vapnik, 1995), es un conjunto de algoritmos de aprendizaje para problemas de clasificación. El algoritmo de aprendizaje o también llamado "machine", consiste en emplear una transformación no lineal a los vectores de entrada (datos de entrada) de modo que estos queden representados en un espacio vectorial de mayores dimensiones. Finalmente, la clasificación se lleva a cabo mediante la definición de un hiperplano (expresado a través de una función lineal en n dimensiones) en dicho espacio vectorial, también llamado "linear decision surface".

La figura 2.7 de (Gutmann y Hyvarinen, 2012) ejemplifica lo anteriormente mencionado. En ella se observa un input en forma de una imagen (número 2), este número es representado, en primer lugar, de forma vectorial y luego dicha representación es llevada a otro espacio vectorial mediante una transformación no lineal. En dicho espacio, existen vectores encargados de construir el hiperplano, estos son los "support vectors", finalmente, con el hiperplano construido es posible clasificar el dato de entrada.

Un hiperplano está definido por:

$$\widetilde{\boldsymbol{w}} \cdot \boldsymbol{x} + w_0 = 0$$

Donde:

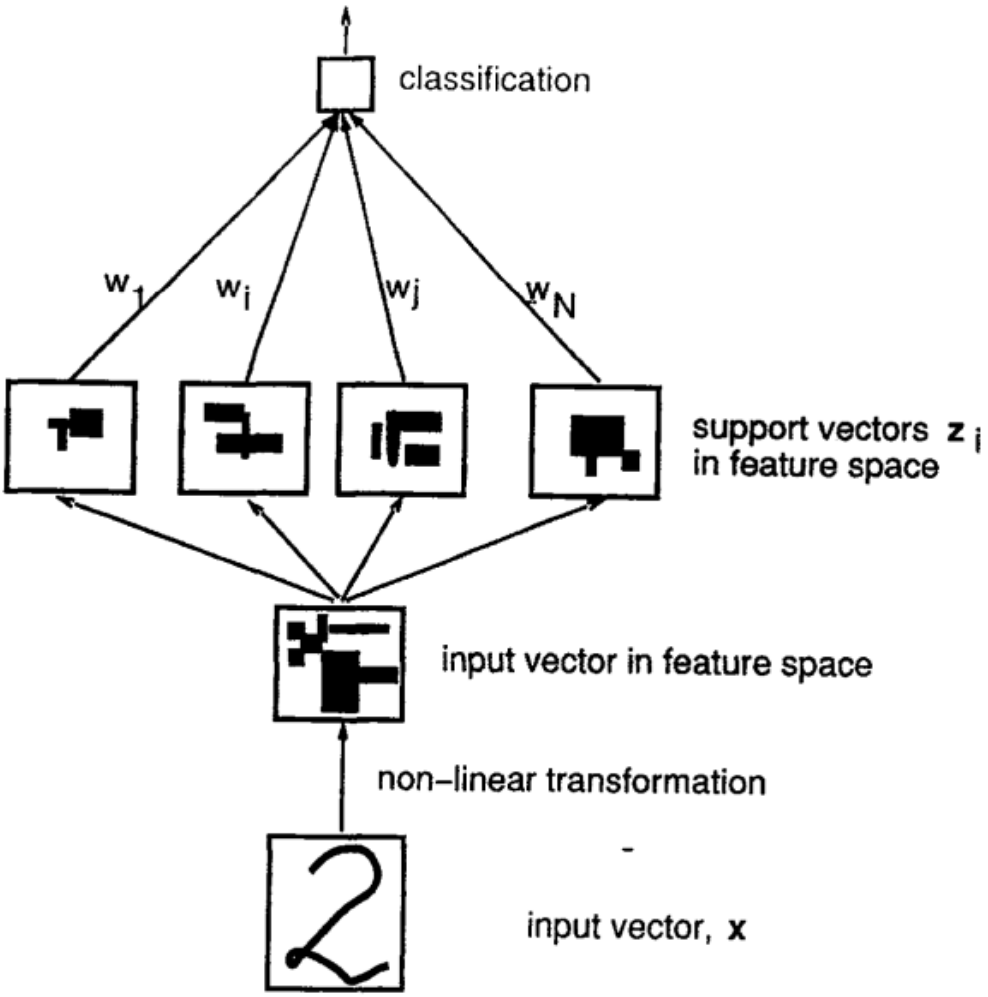
$\boldsymbol{w}$: ($\widetilde{\boldsymbol{w}}$, w_0) un vector de pesos aprendidos donde $w_i \in \mathbb{R}^n$

$\boldsymbol{x}$: Vector de entrada donde $x_i \in \mathbb{R}^n$

n: Dimensión del espacio vectorial

Para problemas linearmente separables, el hiperplano de la ecuación definida arriba representa a un límite que separa entre las dos clases a predecir (clasificación binaria) tal como se muestra en la Figura 2.8 (a) (Williams, 2008).

Figura 2.7 Clasificación de support vector machine.



Fuente: Williams (2008)
Nota: La imagen muestra el funcionamiento de una máquina de soporte vectorial (SVM) para clasificación. El vector de entrada $x$se transforma de forma no lineal a un espacio de características, donde se compara con vectores de soporte. Finalmente, usando pesos w, el modelo decide la clase del dato.

En específico, SVM escoge el vector de pesos $\boldsymbol{w}$ que hace posible la separación de clases en función al "margen máximo" posible (Figura 3(b)).

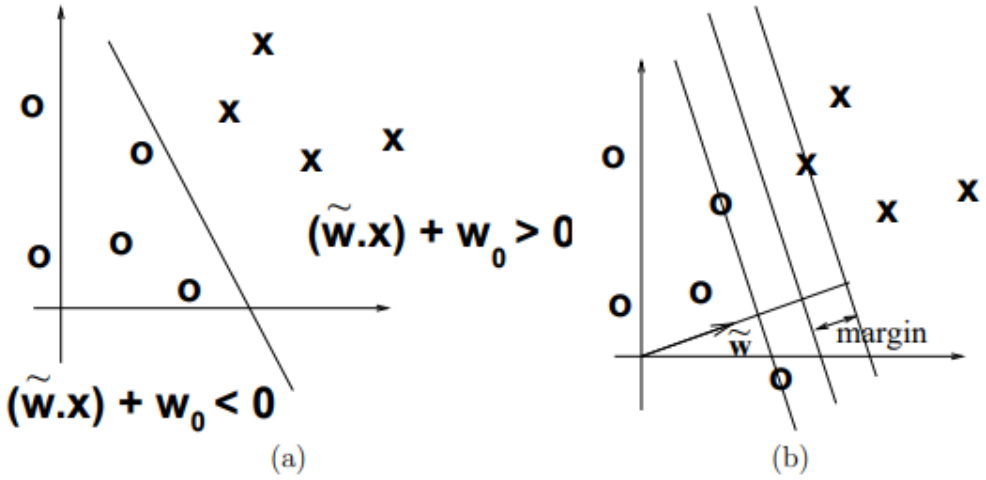
Considerando un set de entrenamiento de la forma $(\boldsymbol{x_i}, y_i)$ para $i = 1, \dots, n$ donde $\boldsymbol{x}$ es el vector de entrada i y $y_i = \pm 1$ es la variable a predecir (label). Dicho margen puede ser calculado como $2/\|\tilde{\boldsymbol{w}}\|$, por lo que para obtener el margen máximo (maximizar el margen), se puede minimizar $\|\tilde{w}\|$ dada la restricción definida por los hiperplanos $y_i(\tilde{\boldsymbol{w}} \cdot \boldsymbol{x_i} + w_0) \geq 1$ (Williams, 2008).

Al ser un problema de optimización, $\tilde{\boldsymbol{w}}$ toma la forma de:

$$\tilde{\boldsymbol{w}} = \sum_i \alpha_i \, y_i \boldsymbol{x_i}$$

Donde α_i son coeficientes determinados mediante métodos numéricos, de modo que se les llama "support vectors" a los vectores $\boldsymbol{x_i}$ con los cuales α_i toma un valor diferente a cero.

Figura 2.8 (a) Hiperplano que separa a dos clases y (b) Hiperplano con margen máximo que separa a dos clases.



Fuente: Williams (2008)
Nota: La imagen muestra cómo un clasificador lineal (SVM) separa dos clases de datos usando una línea o hiperplano definido por $(w \cdot x) + w_0$. Los puntos a cada lado de la línea pertenecen a clases distintas. En la segunda parte se observa el margen, que es la distancia máxima entre la frontera de decisión y los puntos más cercanos de cada clase.

Finalmente, luego del proceso de optimización y por ende entrenamiento, el cálculo de nuevas predicciones para un nuevo input $\boldsymbol{x}$ con SVM, se define como:

$$g(\boldsymbol{x}) = sgn(\widetilde{\boldsymbol{w}} \cdot \boldsymbol{x} + w_0)$$

Cabe resaltar que la función $sgn()$ es empleada ya que se trata de un clasificador binario y por ende los resultados solo toman dos formas $g(\boldsymbol{x}) = \pm 1$.

De forma más general, para un caso donde los puntos en el espacio vectorial no son linealmente separables, se introduce una variable de holgura $\varepsilon_i \geq 0$ por cada registro de entrenamiento, de modo que la función a optimizar es:

$$J = \|\boldsymbol{w}\|^2 + C \sum_{i}^{N} \epsilon_i$$

De acuerdo con Zisserman (2015), la función objetivo de SVM se puede reinterpretar como:

$$\min_{\boldsymbol{w} \in R^d} C \sum_{i}^{N} \max\big(0, 1 - y_i f(x_i)\big) + \|\boldsymbol{w}\|^2$$

Donde:

$\boldsymbol{w} =$ Vector de pesos

$C =$ Parámetro de regularización

$N =$ Número de muestras de entrenamiento

$\boldsymbol{x_i} =$ Registro de entrenamiento

$\boldsymbol{y_i} =$ Ground Truth / Label

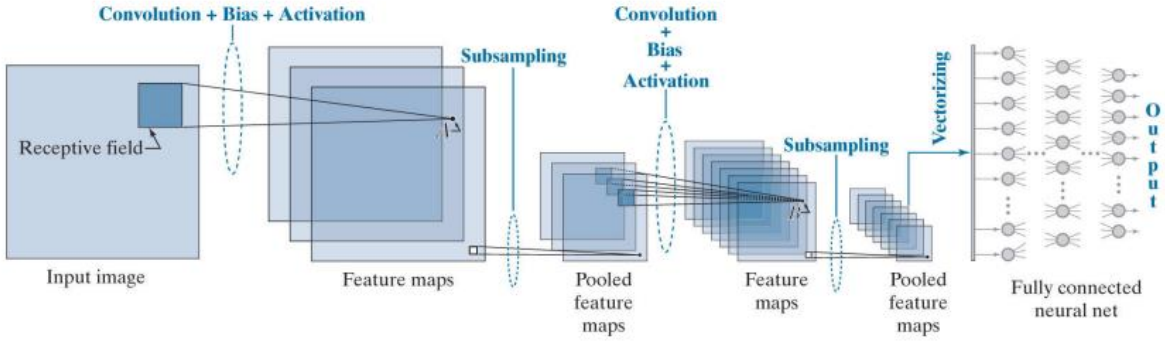
$f(\boldsymbol{x_i}) = \widetilde{\boldsymbol{w}} \cdot \boldsymbol{x_i} + w_0$

2.3.5.2. Redes neuronales Convolucionales

Las Redes Neuronales Convolucionales (también conocidas como ConvNets o CNN en inglés) son un tipo de arquitectura de redes neuronales profundas con la capacidad de identificar y extraer patrones en datos de forma automática (mediante un proceso de entrenamiento). Este tipo de redes usualmente son ideales en tareas de reconocimiento de patrones y clasificación de imágenes propuestas por (Gonzalez y Woods, 2018)

La CNN no es un modelo de clasificación como tal, por lo que usualmente se le agrega una capa final de Perceptrones que se encargan de la clasificación, esta capa también es conocida "fully connected". Un ejemplo de una arquitectura CNN se encuentra en la Figura 2.9 (Gonzalez y Woods, 2018).

Figura 2.9 Arquitectura de la red convolucional LeNet.



Fuente: Lecun et al. (1998)
Nota: Esta imagen ilustra la arquitectura de una Red Neuronal Convolucional (CNN), la cual procesa imágenes mediante capas de convolución y muestreo (*subsampling*) para extraer características visuales. El proceso transforma píxeles en mapas de datos que una red final conectada interpreta para identificar objetos o rostros con precisión.

La figura 2.9 muestra a la arquitectura LeNet (Lecun et al., 1998), una de las primeras redes neuronales convolucionales, cuyo uso fue para clasificar dígitos escritos a mano y recibidos por esta arquitectura a modo de imágenes. *A* y *B* en la figura representan el resultado de aplicar la capa convolucional a los datos de entrada. Asimismo, las arquitecturas CNN usualmente cuentan con capas que reducen la dimensionalidad de los

resultados obtenidos a través de una capa convolucional, estas capas son llamadas *Pooling layers*.

Una gran diferencia entre la red neuronal basada en Perceptrones ("fully connected") y la red neuronal convolucional, es que la última acepta datos de entrada 2D a modo de matriz (usualmente imágenes), mientras que la entrada para la red "fully connected" es unidimensional y representada por vectores. De forma general, el flujo en una arquitectura CNN es muy similar a la de una red neuronal convencional, este flujo consiste en: (1) una suma de productos, (2) añadir el bias a la suma obtenida, (3) evaluar el resultado en una función de activación y (4) el valor obtenido de la función de activación pasa a ser input de la siguiente capa (Gonzalez y Woods, 2018).

La red convolucional no es "fully connected" ya que cada input no conecta con la siguiente capa, en su lugar, primero una vecindad de inputs es procesada, lo que da resultado a un valor que es el que luego se conecta con las siguientes capas, a este proceso se le conoce como Convolución o Convolución espacial (Gonzalez y Woods, 2018). El conjunto de todos los valores resultantes del proceso de convolución forma un *feature map*, formalmente, el proceso es una convolución 2D (en dos dimensiones) que se puede expresar como:

$$w * a_{x,y} = \sum_{l} \sum_{k} w_{l,k} a_{x-l,y-k}$$

Donde:

w: Filtro o kernel de convolución

$a_{x,y}$: Matriz de datos de entrada o matriz resultante de aplicar *Pooling*

$l\ y\ k$: Dimensión del filtro o kernel de convolución

Una capa convolucional está formada por un filtro o kernel que carga con los pesos de entrenamiento, bias y una función de activación. Por lo que un *feature map* z se expresa de la siguiente manera:

$$a = h(z)$$

De modo que:

$$z = w * a_{x,y} + b = \sum_{l} \sum_{k} w_{l,k} a_{x-l,y-k} + b$$

Donde:

$h(x)$: Función de activación

b: Bias

$w * a_{x,y}$: Convolución entre kernel w y $a_{x,y}$

2.3.5.3. Redes Neuronales Convolucionales aplicado a datos textuales

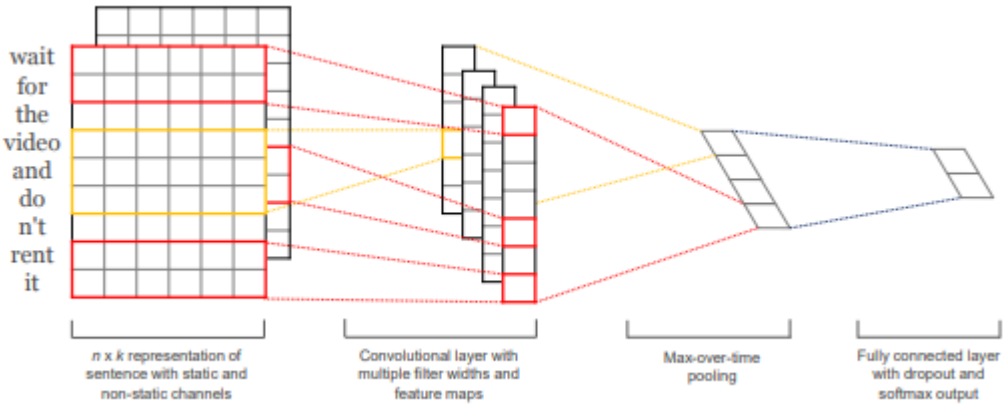
A pesar de haber sido creadas originalmente para tareas del campo de Computer Vision (Kim, 2014) demostró que las CNN pueden extender su uso a tareas no relacionadas al procesamiento y clasificación de imágenes, como, por ejemplo, a tareas del campo del Procesamiento de Lenguaje Natural (NLP). En específico, el autor entrenó una arquitectura basada en CNN con solo una capa convolucional sobre una capa de word embeddings entrenados con Word2Vec.

La figura 2.10 (Kim, 2014)) muestra un ejemplo de la arquitectura usada para CNN en textos. Para el caso de imágenes, estas están representadas por píxeles y por ende es posible formar una matriz a partir de allí, en el caso de textos, estos no pueden representarse como matriz directamente por lo que necesitan pasar por un proceso de vectorización primero. En la figura, la primera capa del modelo es la representación matricial de una oración dada por embeddings, donde cada fila de la matriz es el embedding representativo de una palabra en toda la oración.

Una vez obtenida la matriz de embeddings, el procedimiento convolución es similar al que se realiza con el caso de imágenes, la única gran diferencia es que se aplica una convolución 1D en lugar de una 2D. Por lo que los feature maps resultantes son vectores.

En la figura, a modo de ejemplo, se están aplicando 4 filtros por lo que se tienen 4 feature maps resultantes. Para reducir la dimensionalidad de estos feature maps resultantes se aplica el procedimiento de Pooling a cada feature map de modo que cada uno aporte un valor al vector resultante. Este vector es el resultado de la concatenación de todos los Pooling feature maps que finalmente ingresan a una red neuronal "fully connected" para llevar a cabo la clasificación.

Figura 2.10 CNN con aplicación en datos textuales.



Fuente: Kim (2014)

Nota: Esta imagen muestra una Red Neuronal Convolucional (CNN) aplicada al procesamiento de lenguaje natural, donde las palabras se convierten en vectores numéricos. El sistema utiliza filtros para extraer patrones de texto, realiza un "pooling" para resaltar los datos más relevantes y finalmente clasifica la oración mediante una capa conectada.

Formalmente, el proceso de convolución 1D consiste en:

- Un kernel $\boldsymbol{w} \in R^{hk}$ que es aplicado a una vecindad de h palabras para producir una feature $c_i \in R$.
- Una función de activación f que puede ser lineal o no lineal.

Considerando que una palabra x_i puede estar representada por un embedding de dimensión $n \times k$, es posible concatenar palabras de modo que $\boldsymbol{x_{i:i+j}}$ es un embedding de palabras unidimensional en función de la dimensión k.

Por lo tanto, una feature c_i parte del feature map es generada desde un contexto de palabras $\boldsymbol{x_{i:i+h-1}}$ tal que:

$$c_i = f(\boldsymbol{w} \cdot \boldsymbol{x_{i:i+h-1}} + \boldsymbol{b})$$

Cabe resaltar que la convolución ahora ya no consiste en una doble sumatoria sino solo en el producto punto entre el kernel y el embedding concatenado $\boldsymbol{x}$.

Aplicando el kernel $\boldsymbol{w}$ a cada vecindad de palabras de forma consecutiva es posible producir un feature map $\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-h+1}]$.

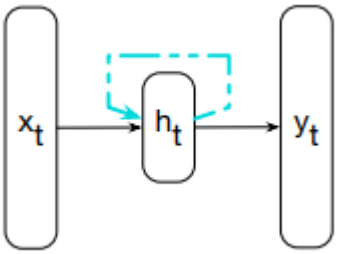
2.3.5.4. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (RNN) son un tipo de red neuronal profunda diseñada para tratar con la naturaleza secuencial y temporal del lenguaje humano. A diferencia de otros modelos neuronales que para realizar sus predicciones usualmente consideran todo un texto al mismo tiempo o a través de ventanas/contextos de palabras, las cuales son independientes entre sí, las RNN son capaces de predecir tomando en cuenta el contexto previo a una palabra, sin necesidad del uso de ventanas/contextos, haciendo uso de sus conexiones recurrentes.

2.3.5.4.1. RNN simple (Elman Network)

Uno de los modelos neuronales que sirvieron para la creación de las bases de las variaciones de RNN más actuales (LSTM o GRU) es la Elman Network propuesto por Elman (1990), una clase de RNN simple que incluye una conexión recurrente en su capa oculta h_t, esta conexión recurrente consiste en el input del modelo x_t y el valor de activación anterior en la capa oculta h_{t-1}. La figura 2.11. ilustra esta arquitectura.

Figura 2.11 Arquitectura de una RNN simple o Elman Network.



Fuente: Elman (1990)
Nota: Esta imagen representa una Red Neuronal Recurrente (RNN), donde la flecha azul cíclica indica que la información persiste a través del tiempo. A diferencia de las redes anteriores, esta utiliza el estado oculto (h_t) para conectar eventos pasados con la entrada actual (x_t), permitiéndole procesar secuencias de datos como texto o video.

En la figura 2.11, se observa como el valor de salida de la capa oculta h_t se conecta consigo misma en el *timestep* $t-1$ y con la entrada x_t. Donde t hace referencia a una variable de tiempo o *timestep*. La capa oculta del timestep anterior cumple una función de memoria que codifica el procedimiento anterior y lo guarda para poderlo informar en algún timestep futuro. La diferencia entre este tipo de RNN y una red neuronal simple, es que la primera agrega una nueva matriz de pesos $\boldsymbol{U}$ que se conecta con la capa oculta del timestep anterior hacia la capa oculta actual (Jurafsky & Martin, 2023).

Formalmente, la RNN simple se representa como:

$$\boldsymbol{h_t} = g(\boldsymbol{U h_{t-1}} + \boldsymbol{W x_t})$$

$$\boldsymbol{y_t} = f(\boldsymbol{V h_t})$$

Donde:

$\boldsymbol{U}$: Pesos de la capa oculta anterior

$\boldsymbol{h_{t-1}}$: Vector de valores de la capa oculta anterior

$\boldsymbol{W}$: Vector de pesos del input

$\boldsymbol{x_t}$: Vector de entrada (input)

$\boldsymbol{V}$: Vector de pesos de la capa oculta actual

$\boldsymbol{h_t}$: Vector de valores de la capa oculta actual

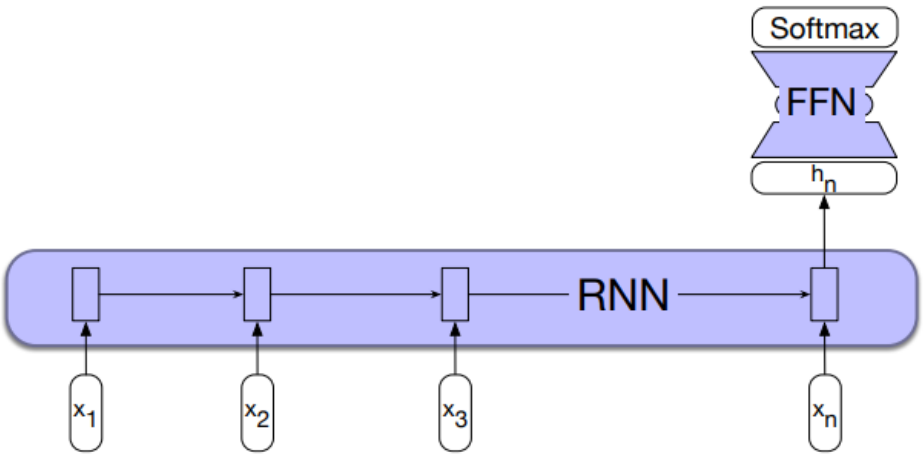
g y f: Funciones de activación

De las ecuaciones se puede deducir que esta versión de RNN necesita entrenar tres tipos de pesos $\boldsymbol{U}$, $\boldsymbol{W}$, $\boldsymbol{V}$, los cuales se mantienen durante todo el proceso de entrenamiento (y timesteps) haciendo posible el uso de *backpropagation*.

La RNN se puede aplicar a diversos tipos de tareas relacionadas a NLP, como, por ejemplo, ser usada como modelo de lenguaje (Mikolov et al., 2010) (language model), de modo que con una RNN se pueda predecir la probabilidad de que cierta palabra w_t ocurra luego de un contexto de palabras dado por $w_{1:t-1}$. Es decir, un modelo de lenguaje neurona les capaz de predecir las palabras siguientes en una oración utilizando métodos de entrenamiento no-supervisados debido a la misma naturaleza secuencial del lenguaje.

Otra aplicación es relacionada a la tarea de clasificación de textos, en este caso, no es necesario predecir las palabras siguientes dada una secuencia de palabras, por lo que usualmente solo se suele anidar una capa softmax de clasificación a la última capa oculta de la RNN tal como se observa en la figura 2.12 (Jurafsky & Martin, 2023).

Figura 2.12 RNN para clasificación de textos.



Fuente: Jurafsky & Martin, 2023)
Nota: Esta imagen muestra una Red Neuronal Recurrente (RNN) desplegada, diseñada para procesar secuencias de datos (x_1 a x_n) de manera consecutiva. El estado final

se procesa a través de una red prealimentada (FFN) y una función Softmax para generar una clasificación o predicción basada en toda la secuencia.

De acuerdo a la figura 2.12, cada token de la secuencia de palabras $x_{1:n}$ (oración) corresponde a un timestep en la RNN representado por embeddings one-hot, de modo que, por ejemplo, para el token $\boldsymbol{x_2}$, la capa oculta $\boldsymbol{h_2}$ se calcula con el vector de entrada proveniente de $\boldsymbol{x_2}$ y la capa oculta anterior $\boldsymbol{h_1}$. Al final, se añade una capa de clasificación Feed Forward Network (FFN) cuya entrada es la capa oculta final de la RNN $\boldsymbol{h_n}$.

2.3.5.4.2. LSTM (Long short-term memory)

Uno de los problemas con la RNN simple descrita anteriormente es la forma en como procesa el contexto de una oración. La información almacenada en los estados ocultos de esta red tiende a ser más relevante para el contexto inmediato de la palabra que se está procesando que para el contexto lejano, esto quiere decir que la RNN simple no es tan eficaz para entender palabras muy lejanas a una cierta palabra (Jurafsky & Martin, 2023).

La Long short-term memory o LSTM (Hochreiter & Schmidhuber, 1997), es una de las variaciones más populares de las RNN, este tipo especial de RNN resuelve el problema de procesamiento del contexto en dos partes: Removiendo información que ya no se necesita del contexto y añadiendo información que podría ser de utilidad para decisiones futuras dentro de la misma red. La LSTM logra esto gracias a que añade una nueva capa de contexto $\boldsymbol{c_t}$ a la red y con el uso de unidades neuronales especializadas que emplean el uso de compuertas o *gates* para controlar el flujo de información que entra y sale de las unidades neuronales.

La LSTM consiste de 3 compuertas, la primera de ellas es la compuerta del olvido (*forget gate*) que se encarga de borrar información que ya no se necesita más del contexto. Se define formalmente como:

$$\boldsymbol{f_t} = \sigma(\boldsymbol{U_f h_{t-1}} + \boldsymbol{W_f x_t})$$

$$\boldsymbol{k_t} = \boldsymbol{c_{t-1}} \odot \boldsymbol{f_t}$$

Por otro lado, la información que sí se necesita y que se quiere extraer de los estados anteriores de la red (contexto) y los inputs actuales se calcula como:

$$\boldsymbol{g_t} = \tanh(\boldsymbol{U_g h_{t-1}} + \boldsymbol{W_g x_t})$$

La siguiente compuerta es la compuerta de adición (*add gate*) que selecciona la información a añadir al contexto actual.

$$\boldsymbol{i_t} = \sigma(\boldsymbol{U_i h_{t-1}} + \boldsymbol{W_i x_t})$$

$$\boldsymbol{j_t} = \boldsymbol{g_t} \odot \boldsymbol{i_t}$$

El nuevo vector de contexto se obtiene:

$$\boldsymbol{c_t} = \boldsymbol{j_t} + \boldsymbol{k_t}$$

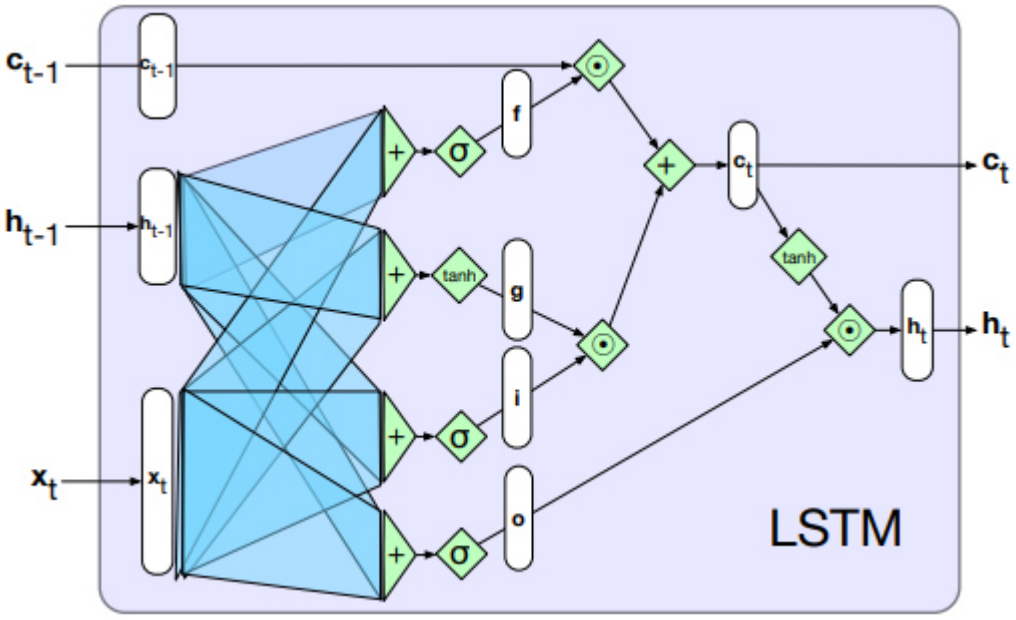
Finalmente, la compuerta de salida (*output gate*) que decide que información se necesita para la capa oculta actual se calcula:

$$\boldsymbol{o_t} = \sigma(\boldsymbol{U_o h_{t-1}} + \boldsymbol{W_o x_t})$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})$$

La figura 2.13 (Hochreiter y Schmidhuber, 1997) muestra una unidad neuronal de una LSTM junto con todo el proceso realizado por las compuertas.

Figura 2.13 Flujo de compuertas en una unidad neuronal de LSTM.



Fuente: Hochreiter y Schmidhuber (1997)
Nota: Esta imagen muestra una celda LSTM (Long Short-Term Memory), un tipo avanzado de red recurrente que utiliza "compuertas" para gestionar el flujo de información. Estas estructuras permiten decidir qué datos olvidar, cuáles guardar y cuáles enviar a la salida, resolviendo el problema de la memoria a corto plazo en secuencias largas.

De la figura 2.13 se desprende que una unidad de LSTM recibe como entradas a un vector input x_t, la capa de contexto del timestep anterior c_{t-1} y la capa oculta del timestep anterior h_{t-1} dando como salida a una capa de contexto c_t y una capa oculta h_t actualizadas. Estas capas actualizadas pueden ser utilizadas como entrada para una nueva unidad de LSTM o de otro modo, la capa oculta h_t puede servir de salida final del modelo LSTM.

2.3.5.5. GRU (Gated Recurrent Unit)

La Gated Recurrent Unit o GRU es una variación del modelo LSTM con menos parámetros, por ende, menos compuertas, en específico, la GRU no posee una compuerta de salida (output gate). Inicialmente propuesta por Cho et al. (2014), la unidad neuronal de una GRU consta de 2 compuertas.

Primero, se calcula la compuerta de reinicio (*reset gate*) que decide si el contexto anterior se ignora o no similar a la *forget gate* de la LSTM. Formalmente está definida como:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

Luego, la compuerta de actualización (*update gate*) controla que tanta información del contexto anterior se va a considerar en el contexto actual, es decir en la capa oculta actual. Donde el contexto actualizado se denota por $\tilde{h}_t$. La update gate se Se define como:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

Finalmente, la activación de la capa oculta h_t se calcula como:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Donde:

$$\tilde{h}_t = g\big(W_h x_t + U_h (r_t \odot h_{t-1})\big)$$

2.3.6. Métricas de evaluación

2.3.6.5. Accuracy

Accuracy es una de las métricas más populares para evaluar el rendimiento de un modelo de Machine Learning/Deep Learning, se calcula directamente de la Matriz de Confusión (Cho et al, (2014):

Figura 2.14 Matriz de Confusión de ejemplo.

		PREDICTED		
	Classes	Positive (1)	Negative (0)	Total
ACTUAL	Positive (1)	TP = 20	FN = 5	25
	Negative (0)	FP = 10	TN = 15	25
	Total	30	20	**50**

Fuente: (Cho et al., 2014)
Nota: Esta matriz de confusión muestra el desempeño de un modelo sobre 50 casos, logrando 35 aciertos (20 Verdaderos Positivos y 15 Verdaderos Negativos). El modelo presenta 15 errores en total: 5 falsos negativos y 10 falsos positivos, lo que permite calcular métricas como la precisión y la sensibilidad.

De modo que:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

TP: True Positives (Verdaderos positivos)

TN: True Negatives (Verdaderos negativos)

FP: False Positives (Falsos Positivos)

FN: False Negatives (Falsos Negativos)

Tomando en cuenta un modelo que predice entre las clases (1) y (0), TP y TN son las predicciones correctamente realizadas por el modelo, es decir, (1) y (0) fueron predichos correctamente. Por otro lado FP y FN son las predicciones incorrectas hechas por el mismo modelo.

Accuracy relaciona todas estas variables dentro de una misma ecuación que, de forma intuitiva, representa la probabilidad de que el modelo prediga correctamente.

La métrica considera que cada registro o unidad contribuye de manera igualitaria al cálculo del Accuracy, por lo que funciona bien para clases balanceadas donde cada registro contribuye igualitariamente a cada clase y por ende el cálculo del error de predición es correcto. Sin embargo, a modo de limitación, en clases inbalanceadas, donde existe mayor cantidad de registros en algunas clases que en otras, Accuracy tiende a asignar un mayor peso a las clases con mayores datos, lo que puede llevar como consecuencia a esconder errores de clasificación graves. (Grandini et al., 2020).

2.3.6.6. F1-Score

F1-Score, similar al Accuracy, mide el rendimiento de clasificación de un modelo predictivo haciendo uso de una matriz de confusión, agregando además a *Precision* y *Recall* bajo el concepto de media armónica.

Donde:

$$Precision_K = \frac{TP_k}{TP_k + FP_k}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k}$$

Debido a que esta métrica está basada en la media armónica, esta tiende a darle mayor peso a menores valores, por ende, F1-Score pondera de manera más equilibrada a clases imbalaceadas donde las clases con menor cantidad de registros también reciben un peso significativo a diferencia que con Accuracy.

En el caso de clasificación de clasificación multiclase, es necesario calcular Precision Y Recall para más de una clase e insertar tales métricas a la media armónica, este proceso lleva a dos diferentes maneras de representar el F1-Score: *F1-Score Micro* y *F1-Score Macro*

2.3.6.7. F1-Score Micro

F1-Score Micro trata a todos los registros como iguales, es decir, asigna el mismo peso a cada registro de manera igualitaria sin tomar en consideración alguna diferencia

existente entre clases. Esto lo realiza calculando micro-Precision y micro-Recall que resultan ser equivalentes para este caso, por ende, la media armónica también toma el mismo valor, de modo que el F1-Score Micro se define como:

$$F1Micro = \frac{\sum_{K=1}^{K} TP_k}{GranTotal}$$

Donde:

$K =$ Número de clases a clasificar

El valor obtenido del F1-Score Micro equivale al del Accuracy, de hecho, son métricas equivalentes y se podría pensar que el F1-Score Micro es el Accuracy multiclase. De hecho, esto también significa que esta métrica comparte las mismas limitaciones del Accuracy, de modo que cada clase tendrá una ponderación diferente, basada en el número de registros existentes en cada una de ellas (Grandini et al., 2020).

2.3.6.8. F1-Score Macro

F1-Macro Score calcula macro-Precision y macro-Recall, promediando a Precision por cada clase predicha y a Recall por cada clase real, respectivamente. De modo que F1-Score Macro asigna la misma ponderación a cada clase al realizar la media armónica, sin importar que existan clases con mayor número de registros que otras.

Macro-Precision y macro-Recall se pueden calcular como:

$$MacroPromedioPrecision = \frac{\sum_{K=1}^{K} Precision_k}{K}$$

$$MacroPromedioRecall = \frac{\sum_{K=1}^{K} Recall_k}{K}$$

De modo que F1-Score Macro es la media armónica entre macro-Precision y macro-Recall:

$$2 \times (\frac{MacroPromedioPrecision \times MacroPromedioRecall}{MacroPromedioPrecision^{-1} \times MacroPromedioRecall^{-1}})$$

De forma intuitiva, con F1-Score Macro, el efecto de clases grandes (con muchos elementos) tiene la misma importancia que clases pequeñas. De esta forma, valores altos de F1-Score Macro indican que el modelo predictivo tiene un buen rendimiento de clasificación en todas las clases, mientras que el caso contrario indica que el modelo tiene un rendimiento pobre en todas las clases (Grandini M. et al., 2020).

2.3.7. Herramientas y librerías

2.3.7.5. Python

Para la presente investigación se ha usado el lenguaje de programación Python, el cual es un lenguaje de alto nivel interpretado cuya filosofía principal se sostiene en la legibilidad de su código. De acuerdo con (Tuttle, L., 2025), Python es la opción predilecta para el tratamiento de datos y desarrollo de aplicaciones en IA, es considerado como el lenguaje de programación del futuro. Su versatilidad para el desarrollo de aplicaciones web y trabajos colaborativos lo han hecho muy atractivo para su uso. Adicionalmente, la composición de librerías para el análisis y visualización de datos lo han catapultado como herramienta indispensable para aplicaciones en ciencia de datos e inteligencia artificial. Junto con Python se han utilizado plataformas de desarrollo como Google Colab versión limitada y de pago para procesos de entrenamiento de modelos que demandaron alta capacidad computacinal y Jupyter notebook de Anaconda.

2.3.7.2. Twint

Es una herramienta de scraping y de Open Source INTelligence (OSINT) para la red social Twitter desarrollada en el lenguaje de programación Python. Su principal característica diferenciadora (a comparación de otras herramientas de scraping para Twitter), es que esta no emplea la API oficial de Twitter para realizar el procedimiento de extracción de datos y por ende no tiene las mismas limitaciones de otras herramientas que sí emplean la API oficial, esto se traduce en un procedimiento de extracción más veloz y con mayor cantidad de datos obtenidos.

Esta herramienta fue desarrollada por el proyecto “twintproject” y se encuentra disponible en github (Twintproject, 2021)

Beneficios:

- Se pueden recolectar casi todos los tweets disponibles de un perfil a diferencia que con la API oficial.
- Se puede utilizar sin cuenta de Twitter.
- No tiene limitaciones por tiempo de extracción.
- Permite recolectar datos de usuarios específicos, hashtags y usar las funciones de la búsqueda avanzada de Twitter
- Los datos se pueden exportar en una variedad de formatos incluido CSV.

2.3.7.3. Matplotlib

Es una librería de visualización muy famosa disponible para el lenguaje de programación Python, permite la creación de diversos tipos de visualizaciones entre estáticas, dinámicas e interactivas. Además, viene embebida en entornos de programación por notebooks como JupyterLab (Hunter, 2007). La librería matplotlib se encuentra disponible en la web (Matplotlib, 2021)

2.3.7.4. Seaborn

Es una librería de visualización de datos basada en Matplotlib, a diferencia de esta última, Seaborn ofrece un manejo de gráficos bajo una interfaz de alto nivel, es decir, con un mayor nivel de abstracción que permite al usuario crear visualizaciones con mayor facilidad.

Seaborn necesita de un dataset y una especificación del gráfico a realizar, con estos dos ítems correctamente cargados, la librería automáticamente transforma los datos a atributos visuales que comprenden el color, tamaño, estilo, internamente calcula funciones de estadística descriptiva y decora el gráfico con etiquetas y leyendas (Waskon, 2021). El servicio seabor se encuentra disponible en la web (Seaborn, 2021)

2.3.7.5. NLTK

Natural Language Toolkit (NLTK) es una de las plataformas líderes para construir programas en Python que trabajan con datos de lenguaje natural. La librería provee una interfaz de alto nivel que permite acceder a alrededor de 50 recursos léxicos como WordNet, junto con una variedad de librerías de procesamiento de texto para clasificación, tokenización, stemming, tagging, etc. NLTK es usado en diversos contextos profesionales que incluyen ingenieros, investigadores, linguistas, entre otros, y su fácil acceso a sus herramientas lo hace atractivo para estudiantes y educadores también. La librería es open-source y está disponible para Windows, Mac OS X y Linux. (Bird et al. 2009). El servicio de nltk está disponible en la web (nltk, 2021)

2.3.7.6. Gensim

"Generate Similar" (Gensim) es una librería open-source de Python para representar documentos textuales en forma de vectores semánticos de forma eficiente y a través de una interfaz amigable con el usuario.

Gensim está diseñado para trabajar con un input textual a modo de texto plano para ser procesado por algoritmos no supervisados. Los algoritmos no supervisados más populares disponibles en la librería incluyen a Word2Vec, FastText, LSI, LDA, los cuales se encargan de descubrir features o características de los textos sin necesidad de un input humano. La librería está diseñada y construida desde cero siguiendo tres pilares fundamentales: Práctica (Uso sencillo e intuitivo), Independencia de memoria (No hay necesidad de cargar todo el dataset en RAM) y Rendimiento (Implementaciones de algoritmos altamente optimizadas con C) (Radim y Sojka (2010)). El servicio de gensim está disponible en la web (Gensim, 2021)

2.3.7.7. Scikit-Learn

Scikit-Learn es una librería de Python que integra un amplio rango de algoritmos de Machine Learning para problemas de media escala ya sean supervisados o no supervisados. El módulo está enfocado en brindar herramientas de Machine Learning a personas no especializadas en el campo a través de un lenguaje de programación de alto nivel. El proyecto como tal tiene un énfasis constante en la facilidad de uso, rendimiento,

documentación adecuada y consistencia en el diseño, así como, el empleo de dependencias mínimas (Pedregosa et al., 2011).

La librería ofrece distintos tipos de estimadores o clasificadores basados en algoritmos de Machine Learning, transformadores o preprocesadores para el preprocesamiento de los datos, un interfaz para la creación de Pipelines que combinen a un conjunto de transformadores y estimadores, así como un módulo para la evaluación de modelos ya sea mediante Cross Validation o una versión más robusta que utiliza una estructura de datos grilla para la búsqueda de hiperparámetros (Grid Search). El servicio Scikit-Learn se en encuentra en la web (Scikit-Learn, 2021)

2.3.7.8. Tensorflow

Tensorflow es una plataforma open-source para Machine Learning y Deep Learning. Comprende todo un ecosistema de herramientas, librerías y módulos que hacen posible el desarrollo de modelos neuronales flexibles y de fácil despliegue, así como también permite a investigadores implementar experimentos con modelos que desafíen al estado del arte.

Tensorflow fue desarrollado originalmente por investigadores e ingenieros parte del equipo de Machine Intelligence de Google Brain para realizar investigaciones en Machine Learning y redes neuronales. La librería está disponible para Python y C++ mediante APIs estables y con mantenimiento continuo.

Tensorflow ofrece distintos niveles de abstracción, en el más alto nivel se encuentra la API llamada Keras que facilita la implementación de modelos de redes neuronales, también cuenta con abstracciones para sistemas distribuidos a través de la API de Distribution Strategy. Su filosofía de desarrollo permite que todos los modelos implementados en Tensorflow sean de fácil despliegue en cualquier tipo de dispositivo, incluyendo Desktop y Mobile.

Aunque su uso es global para todo tipo de algoritmos de Machine Learning, Tensorflow tiene un foco importante en Deep Learning, ya que provee implementaciones eficientes de algoritmos basados en redes neuronales profundas como CNN, GRU, BERT, etc y que pueden ser entrenados en diferentes tipos de arquitecturas como CPU o GPU sin

la necesidad de hacer un cambio drástico al código. Servicio Tensorflow dispuesto en la web (Tensorflow, 2021).

2.3.7.9. PyMuPDF

PyMuPDF está basado en la librería MuPDF, una librería ligera para cargar, renderizar y manejar a través de diversas herramientas archivos de tipo PDF y XPS, EPUB, etc, MuPDF es reconocida por ser una librería de alto rendimiento.

PyMuPDF es una abstracción de MuPDF para el lenguaje de programación Python, esta librería hereda todas las características y virtudes de MuPDF y adiciona el soporta para imágenes en los formatos más populares como .jpg, .png, .bmp, etc.

Esta librería permite desencriptar y cargar documentos, acceder a metadata, renderizar documentos (ej. una página en pdf a png o svg), buscar un texto en el documento, extraer texto e imágenes, realizar OCR (Optical Character Recognition) usando Tesseract. Especificamente para archivos PDF, PyMuPDF permite realizar una variedad de acciones como hacer split de páginas, unir documentos, unir páginas, insertar texto o imágenes, etc.

La librería usada es PyMuPDF y es open-source y está disponible en la web (Github, 2021)

CAPÍTULO 3 METODOLOGÍA

3.1. Hipótesis general

El framework basado en técnicas de Procesamiento de Lenguaje Natural y Machine Learning permite la identificación de tópicos de interés y clasificación en lo expresado en redes sociales, planes de gobierno de partidos políticos y lo expuesto en el debate presidencial para procesos de predicción de clasificación en el contexto de elección presidencial 2021.

3.2. Hipótesis específica

HE1: El framework para el análisis de lo expresado en redes sociales, planes de gobierno y debate presidencial, sirve de referencia para procesos procesamiento, clasificación y predicción de post en el contexto de la elección presidencial 2021 mediante técnicas de Procesamiento de Lenguaje Natural y Machine Learning.

HE2: Las técnicas de Procesamiento de Lenguaje Natural posibilitan el procesamiento de posteos en redes sociales, análisis de planes de gobierno y transcripciones del debate presidencial 2021

HE3: Las técnicas de Machine Learning posiblitan el procesamiento de tópicos de interés de data proveniente de redes sociales, planes de gobierno y el debate presidencial 2021.

HE4: Los algoritmos de Machine Learning y respectivamente las métricas de evaluación permiten predecir de forma efectiva post de proselitismo político de redes sociales

3.3. Diseño de la investigación

3.3.1. Diseño

La investigación es exploratoria en vista que en el proceso se identificará los tópicos de interés y su clasificación.

La investigación es transversal al realizarse en el contexto de un periodo específico del tiempo.

3.3.2. Tipo

La investigación es exploratoria cuantitativa, en vista que el tratamiento de las variables y su interpretación es de corte numérica.

3.3.3. Enfoque

El enfoque de la investigación es cuantitativo debido a que los resultados numéricos en las diversas etapas de la investigación permitirán por ejemplo definir la selección de los algoritmos idóneos para el procesamiento como para la predicción.

3.3.4. Población y muestra

La población de estudio corresponde a:

- Todos los posts de la red social Twitter correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.
- Todos los planes de gobierno correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.
- Todas las trascripciones del debate presidencial correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.

3.3.5. Operacionalización de Variables

Tabla 3.1: Operacionalización de las variables

Variable	Indicador	Cálculo/Valores
Efectividad del modelo	Precisión (Accuracy/F1 score micro) (Grandini et al., 2020)	$$\frac{\sum_{K=1}^{K} TP_k}{\sum_{K=1}^{K} TotalColumna_k} = \frac{\sum_{K=1}^{K} TP_k}{GranTotal}$$ Donde: K =Número de clases a clasificar TP =True positives (Registros predichos correctamente por el modelo)
	F1 score macro (Grandini et al., 2020)	$$2 \times \left(\frac{MacroPromedioPrecision \times MacroPromedioRecall}{MacroPromedioPrecision^{-1} \times MacroPromedioRecall^{-1}}\right)$$ Se calculan los promedios de Precision y Recall por cada clase K: $$MacroPromedioPrecision = \frac{\sum_{K=1}^{K} Precision_k}{K}$$ $$MacroPromedioRecall = \frac{\sum_{K=1}^{K} Recall_k}{K}$$ Donde: $$Precision_K = \frac{TP_k}{TP_k + FP_k}$$ $$Recall_k = \frac{TP_k}{TP_k + FN_k}$$ K =Número de clases a clasificar TP =True positives (Registros predichos correctamente por el modelo) FP =False positives FN =True negatives

Variable	Indicador	Cálculo/Valores
Métodos de representación de texto	TF-IDF	$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$ Donde: $tf_{i,j}$ = Número de ocurrencias de i en j df_i = Número de documentos que contiene i N = Número total de documentos
	Word embeddings (Word2Vec y FastText Skip-gram)	Basado en Mikolov et al. (2013), la función de costo a maximizar es el promedio de la probabilidad logarítmica: $$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log(p(w_{t+j}\|w_t))$$ Donde: T = Tamaño del corpus c = Tamaño del contexto (window) de entrenamiento w_t = Palabra central/fijada, dada la palabra central, se quiere encontrar la probabilidad de que otra palabra w_{t+j} se encuentre dentro del contexto de entrenamiento delimitado por $[-c, c]$. j = Índice que se añade a la ubicación de la palabra central w_t dentro del contexto de entrenamiento. FastText (Bojanowski & Olivera, 2016) está basado en lo propuesto por (Mikolov et al., 2013), solo que esta vez en lugar de considerar el score entre dos vectores w y v (input y ouput vectors), se obtiene un score agregado entre los n-grams de la palabra representada por w con el output vector v: $$s(w,c) = \sum_{g\in G_w} z_g v_c$$ Donde: z_g = Vector que representa al n-gram g que aparecen en la palabra w v_c = Vector output también conocido como vector contexto.

		g = n-gram g basado en la palabra w G_w = Conjunto de n-grams que aparecen en w
Variable	**Indicador**	**Cálculo/Valores**
Modelo predictivo	Support Vector Machine (SVM)	Consiste en minimizar la función loss (Zisserman (2015)): $$\min_{w \in R^d} C \sum_{i}^{N} \max\left(0, 1 - y_i f(x_i)\right) + \|w\|^2$$ Donde: w = Weight vector / vector de pesos C = Parámetro de regularización N = Número de muestras de entrenamiento x_i = Registro de entrenamiento y_i = Ground Truth / Label Y además: $$f(x_i) = w^T x_i + b$$ Donde: b = Término de bias/sesgo
	Redes neuronales convolucionales	El proceso de convolución 1D para clasificación de textos tal como está definido por Kim (2014) consta de: - Un kernel $w \in R^{hk}$ que es aplicado a un contexto/ventana de h palabras para producir una feature $c_i \in R$. - Una función de activación f que puede ser lineal o no lineal. Considerando que una palabra x_i puede estar representada por un vector embebido de dimensión $n \times k$, es posible concatenar palabras de modo que $x_{i:i+j}$ es un vector de palabras unidimensional en función de la dimensión k. Por lo tanto, una feature c_i es generada desde un contexto de palabras $x_{i:i+h-1}$: $$c_i = f(w \cdot x_{i:i+h-1} + b)$$ Donde: b = Bias $\in R$

		Aplicando w a cada ventana de palabras de forma consecutiva es posible producir un feature map $\boldsymbol{c} = [c_1, c_2, \dots, c_{n-h+1}]$
Variable	**Indicador**	**Cálculo/Valores**
Modelo predictivo	Redes neuronales recurrentes (GRU)	Define por Dey y Salem (2017): $$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$ $$\tilde{h}_t = g(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$ Y dos compuertas (gates): $$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$ $$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$ Donde: x_t = Vector input / registro de entrenamiento h_t = Output vector / Celda de memoria generada $\tilde{h}_t$ = Vector candidato intermedio para la celda de memoria interna z_t = Update gate / Celda de actualización r_t = Reset gate / Celda de reinicio W, U, b = Matrices y vectores de pesos/parámetros a entrenar respectivamente. σ = Función de activación puede ser lineal o no lineal
Fuente de información	Base de datos	a. Twitter de partidos políticos en contienda en proceso de elección presidencial peruano 2021 b. Planes de gobierno (o equivalente) de los partidos políticos en contienda en proceso de elección presidencial peruano 2021 c. Transcripciones del debate presidencial de los candidatos de los partidos políticos en contienda en proceso de elección presidencial peruano 2021

Fuente: Elaboración propia

Nota: La matriz operacionaliza la evaluación de modelos predictivos (SVM, CNN, GRU) aplicados al análisis de textos políticos peruanos mediante métodos de representación como TF-IDF y Word Embeddings, tomando como fuente Twitter, planes de gobierno y debates presidenciales de las elecciones de 2021.

Para los modelos de Deep Learning (CNN y GRU) se utilizó la función loss Categorical Cross Entropy para clasificación multiclase (Ciampiconi et al., 2023):

$$L_{CCE} = -\frac{1}{K}\sum_{j=1}^{K} \log(\widehat{f_k}(x))$$

Donde:

$K =$ Número de clases a clasificar

$f_k(x) =$ Score predicho por el modelo para cada clase K

$\widehat{f_k}(x) = f_S(f_K(x))$ donde $f_s(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}$ es la función softmax

3.3.6. Matriz de Consistencia

Tabla 3.2: Operacionalización de las variables

PROBLEMAS ESPECÍFICOS	OBJETIVOS ESPECÍFICOS	HIPÓTESIS	VARIABLES	DIMENSIÓN	INDICADORES
PE1. ¿De qué manera la propuesta de un framework para el análisis de lo expresado en redes sociales, planes de gobierno y debate presidencial, posibilitará servir de referencia para procesos de clasificación y predicción de post en el contexto de la elección presidencial 2021 que incluyan técnicas de Procesamiento Natural y técnicas de Machine Learning?	OE1. Proponer un framework basado en PLN y ML para el análisis de redes sociales, planes de gobierno y debates presidenciales	La propuesta de un framework basado en PLN y ML permite estructurar de manera sistemática el análisis del discurso político digital	Framework metodológico	Diseño metodológico	Definición de fases, modularidad, replicabilidad
PE2: ¿Qué técnicas de Procesamiento de Lenguaje Natural posibilitará el procesamiento de posteos en redes sociales, análisis de planes de gobierno y transcripciones del debate presidencial 2021?	OE2. Identificar técnicas de PLN para el procesamiento de posteos, planes de gobierno y debates	Las técnicas de PLN permiten transformar datos textuales no estructurados en representaciones analizables	Procesamiento de Lenguaje Natural	Preprocesamiento textual	Limpieza, tokenización, normalización, extracción de características
PE3: ¿Qué técnicas de Machine Learning serán las pertinentes para el procesamiento de	OE3. Identificar técnicas de ML para el análisis de	Las técnicas de ML permiten identificar	Machine Learning	Modelado y análisis	Clasificación de tópicos, precisión del modelo

tópicos de interés de data proveniente de redes sociales, planes de gobierno y el debate presidencial 2021?	los tópicos de interés	patrones discursivos relevantes en el discurso político			
PE4: ¿Qué algoritmo de Machine Learning y respectivamente métricas de evaluación permitirá la predicción de clasificación de post de proselitismo político en redes sociales?	OE4. Identificar algoritmos y métricas para la predicción de clasificación de mensajes políticos	El uso de algoritmos y métricas adecuadas mejora la confiabilidad de la clasificación de mensajes políticos	Clasificación predictiva	Evaluación del modelo	Accuracy, F1-score, F1-score micro

Fuente: Elaboración propia

3.4. Tipo y diseño de investigación

3.4.1. Tipo de investigación

El tipo de investigación es exploratoria en vista que en el proceso se identificará los tópicos de interés y su clasificación. Así mismo es cuantitativa debido a que se busca la determinación de un score óptimo para procesos predictivos, que forma parte del proceso de investigación. Es una investigación transversal al realizarse en un mismo periodo sin depender de cambios en el tiempo.

3.4.2. Enfoque

El enfoque de la investigación es cuantitativa en vista que el los resultados se medirán en función al tratamiento de variable numéricas y métodos computacionales que brindan soporte interpretativo.

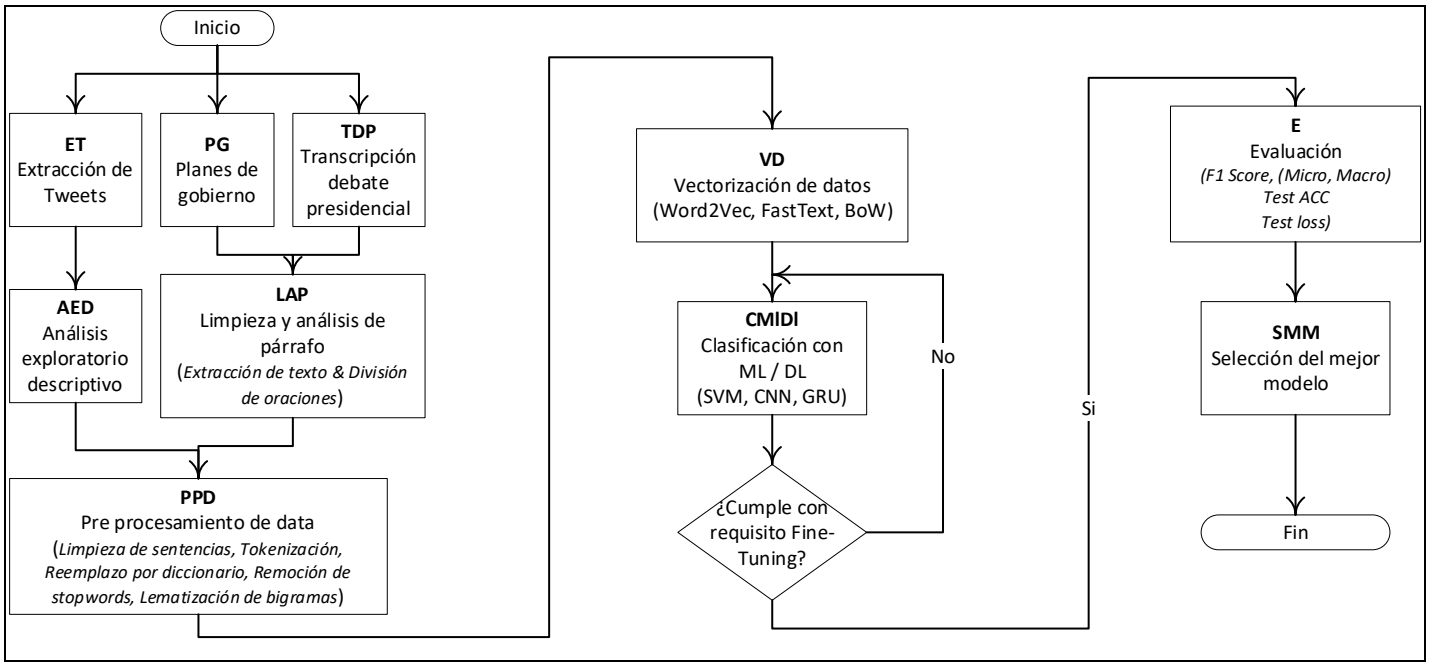
3.4.3. Población y muestra

La población de estudio corresponde a:

- Todos los posts de la red social Twitter correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.
- Todos los planes de gobierno correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.
- Todas las trascripciones del debate presidencial correspondiente a todos los candidatos y/o partidos políticos participantes en la contienda de elección presidencial 2021.

3.4.4. Metodología de implementación

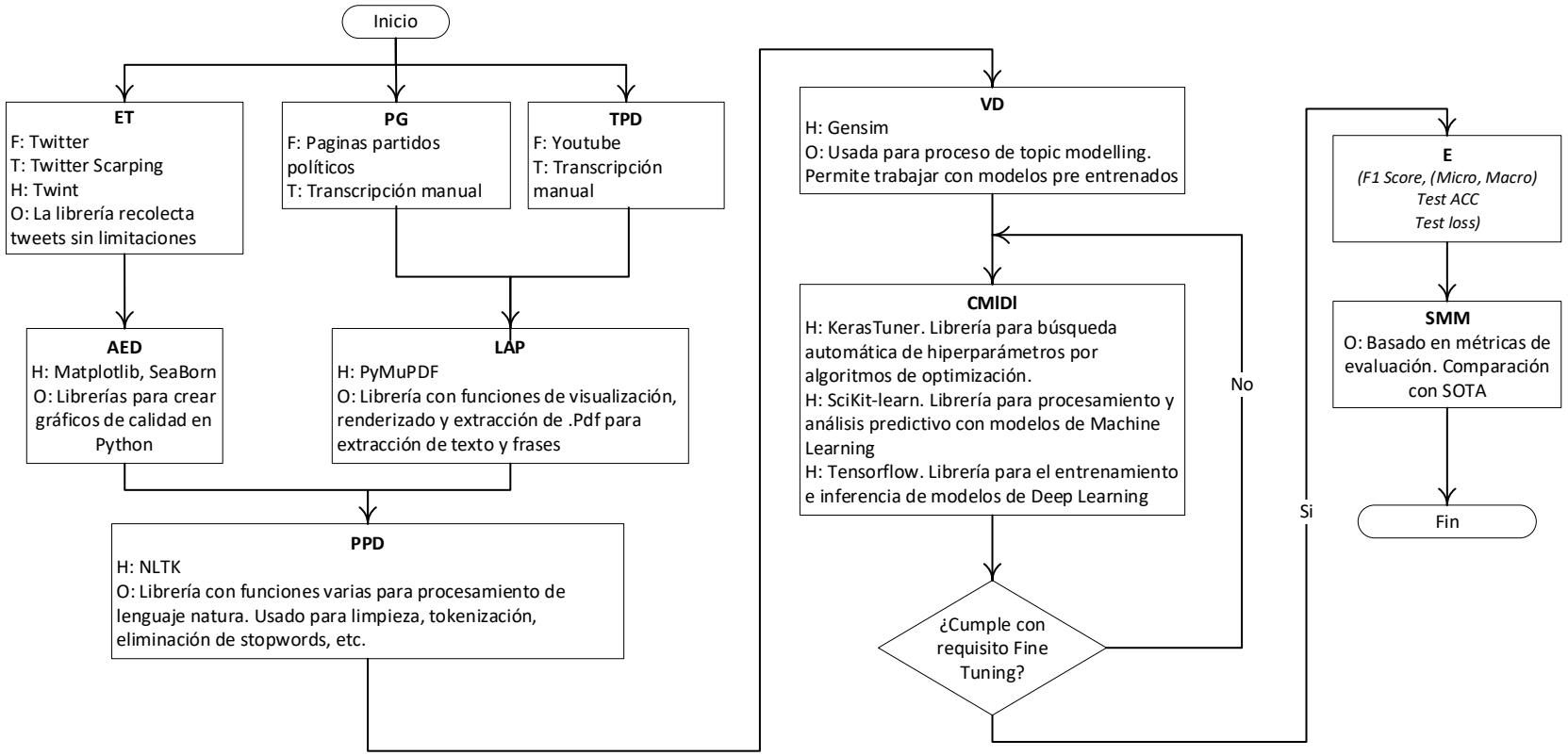
Figura 3.1Framework metodológico.



Fuente: Elaboración propia

Nota: El diagrama de flujo expuesto, detalla el proceso sistemático de la investigación emprendida, desde la extracción de textos hasta la selección del mejor modelo predictivo. Describe etapas clave como el preprocesamiento de data, la vectorización mediante algoritmos como Word2Vec y la clasificación con modelos de Machine Learning y Deep Learning (SVM, CNN, GRU). Finalmente, el proceso concluye con una evaluación exhaustiva basada en métricas de precisión y pérdida para determinar la eficacia de la solución.

Figura 3.2 Framework metodológico con detalle de técnicas y herramientas usadas



F: Fuente
T: Técnica

H: Herramientas
O: Observaciones

Fuente: Elaboración propia

Nota: El diagrama de flujo expuesto, es una variante del diagrama de flujo expuesto en la figura 3.1. En este diagrama, se hace alusión a las técnicas usadas en cada etapa y alusiones a las fuentes de datos.

3.5. Aplicación metodológica

La metodología propuesta destaca todo el proceso analítico que va desde la extracción de datos hasta la selección del modelo predictor ideal. Para concretar este framework se ha revisado otras propuestas metodológicas con similares problemáticas y objetivos.

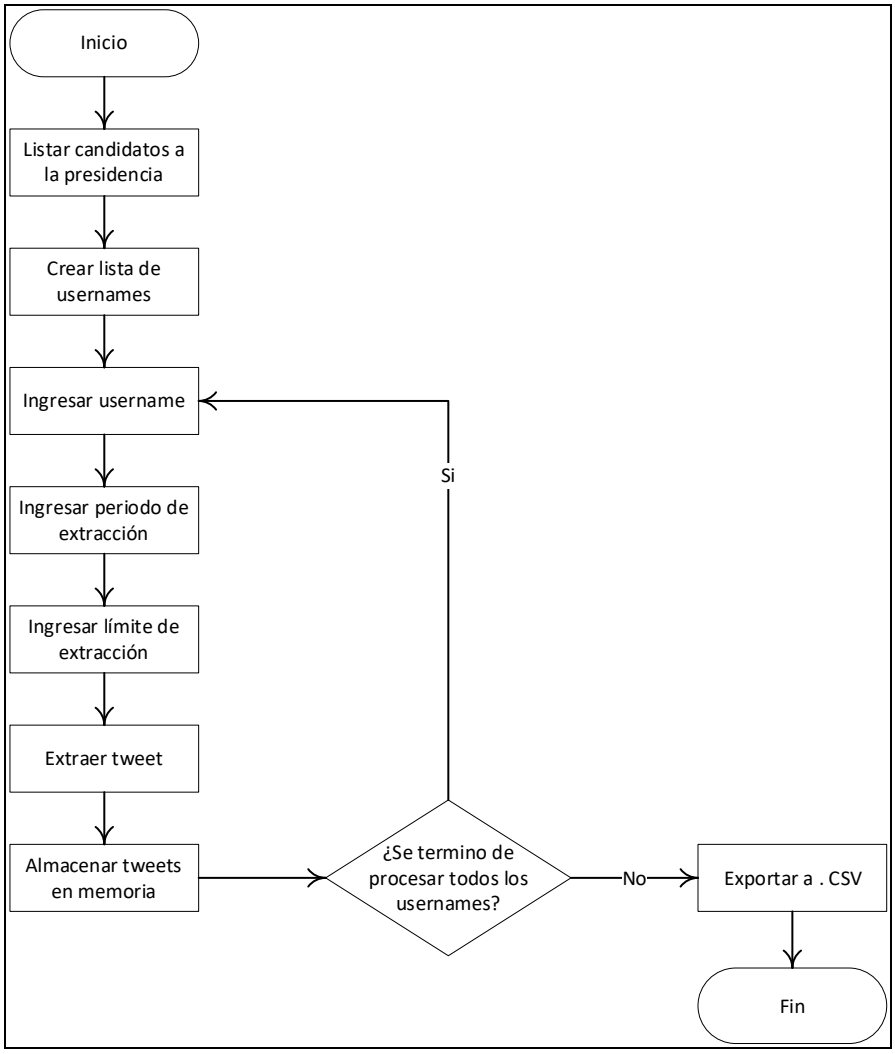
3.5.1. Extracción de datos

Dado el objetivo de investigación, las fuentes de los datos para la investigación fueron:

a. Post de cuentas de Twitter de todos los partidos contendientes en el proceso de elección presidencial 2021.
b. Planes de gobierno o equivalente rescatado desde páginas oficiales de cada partido político participantes en el proceso de elección presidencial 2021.
c. Transcripción de los debates presidenciales auspiciados por JNE los cuales fueron transmitidos por señal abierta.

En la figura 3.3 se expone el proceso algorítmico para la extracción de Tweets se realizó usando una librería especializada en scrapping para Twitter, twint, en ella se especificó realizar la búsqueda por nombre de usuario de twitter (Username), periodo de extracción (Since, Until), así como límite máximo de tweets a extraer (Limit). Cabe resaltar que la lista de candidatos a la presidencia, así como sus usuarios fueron identificados y prcesados de forma manual.

Figura 3.3 Algoritmo para la extracción de tweets de candidatos a la presidencia usando twint



Fuente: Elaboración propia
Nota: El diagrama de flujo describe el proceso de extracción de datos desde Twitter. Para ello, se parte de una lista previamente identificada de candidatos y sus respectivas cuentas. A través de la aplicación Twint se realiza la extracción de los comentarios (tweets), los cuales posteriormente son almacenados en archivos CSV y preparados para su procesamiento.

Inputs: Usernames de twitter, period de extracción (Since, Until), límite de extracción

Outputs: Archivos de datos .csv por cada username eg. Si hay 3 candidatos, se obtienen 3 archivos .csv

En la figura 3.4 se muestra el código usado para la extracción de Tweets usando la librería twint; en el código se especifican los parámetros de extracción ya descritos (Username, Since, Until y Limit), así como el parámetro Output en donde se especifica la ruta de exportación del archivo .csv que contiene los tweets ya extraídos. Los demás parámetros se dejan por defecto.

Figura 3.4 Código Fuente para la extracción de tweets de candidatos a la presidencia usando twint

scrapingTweets.py 2

D: > Users > jmaldonado > OneDrive - Universidad ESAN > Investigacion > scraping_twitter > scrapingTweets.py > ...

```
import twint
import nest_asyncio

nest_asyncio.apply()

c = twint.Config()

#c.Search = "gustavopetro OR petrogustavo OR rodolfohernandez OR #colombiavota2022 OR #eleccionescolombia
c.Username = "Vero_Mendoza_F"
c.Custom["tweet"] = ["id", "created_at", "tweet", "username"]
c.Since = "2021-06-01"
c.Until = "2021-12-18"
c.Limit = 100000
#output
c.Hide_output = False
c.Store_csv = True
c.Output = "corpus_peru_candidatos/@"+c.Username+".csv"

twint.run.Search(c)
```

Fuente: Elaboración propia
Nota: El código fuente expuesto, detalla los parámetros a ingresar en la aplicación Twint para obtener ID, fecha, contenido del tweet y username dentro de un período determinado y con un límite de resultados.

El proceso de extracción usando el código se repite hasta agotar la lista de usernames de twitter previamente recolectados (Se obtendrá un archivo .csv por cada username y cada archivo .csv puede contener hasta 100,000 tweets). Para el caso del ejemplo en la figura 3.4, se especificó que la carpeta de salida de los tweets en .csv sea en "corpus_peru_candidatos".

En el anexo 1 se detalla el código Python para el proceso de scraping de tweets.

El proceso de anotación, fue realizado por estudiantes de pregrado de ciencias sociales. Luego, se realizó un proceso de comparación entre anotaciones para encontrar el

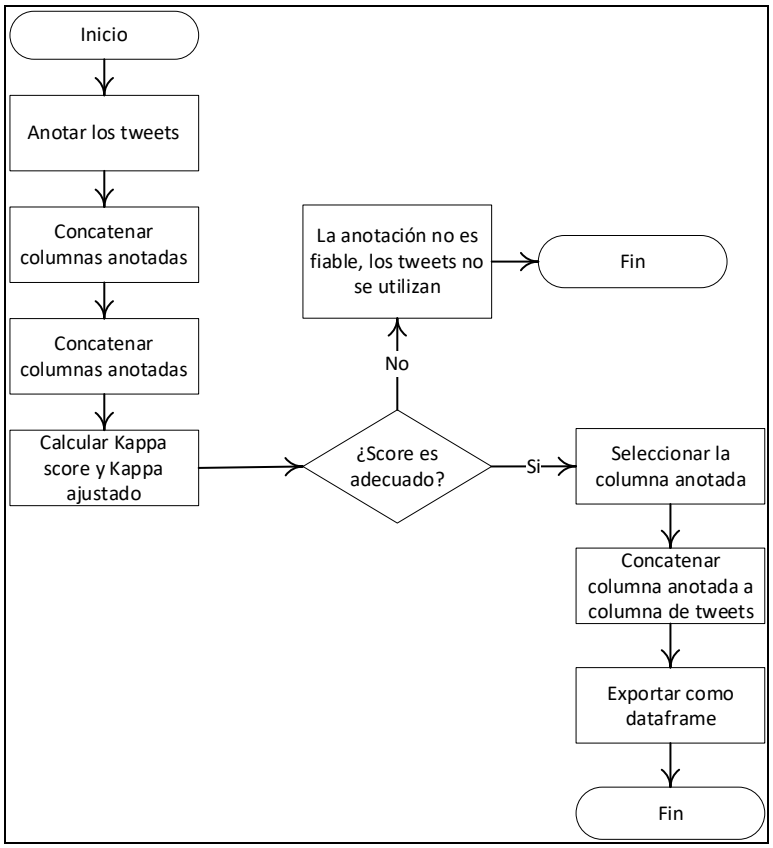
grado de acuerdo entre anotadores, para finalmente concluir si es posible o no usar las anotaciones como fuente fiable.

El procedimiento es descrito en el diagrama de flujo de la figura 3.5

Inputs: Tweets de candidatos presidenciales previamente extraidos

Outputs: Archivo dataframe con los tweets correctamente anotados

Figura 3.5 Proceso para el análisis y anotación de los tweets extraídos



Fuente: Elaboración propia
Nota: El diagrama de flujo ilustra el proceso de anotación de los tweets, donde se concatenan las columnas de resultados para calcular los coeficientes **Kappa** y **Kappa ajustado**. Este procedimiento permite evaluar de manera cuantitativa el nivel de concordancia entre los anotadores.

En el anexo 2 se detalla el código Python para el proceso de análisis y anotación de tweets.

Sobre el proceso de recolección de planes de gobierno se identificó las páginas oficiales de los partidos politicos y se descargo los documentos en formato .pdf.

Input: Sitio web del partido político

Output: Archivo en pdf del plan de gobierno del partido político

Sobre el proceso de transcripción de los debates politicos, utilizó como fuente la plataforma youtube, donde diversos canals publicaron enteramente los debates presidenciales de principio a fin. El proceso de transcripción fue realizado de forma manual con la ayuda de 3 estudiantes de pregrado. El resultado fue un archivo de texto en .pdf con la transcripción textual de cada debate presidencial, la transcripción contenía al emisor de cada frase y también estaba organizada por temas políticos debatidos.

Input: Debate televisado en video

Output: Archivo pdf con las transcripciones del debate

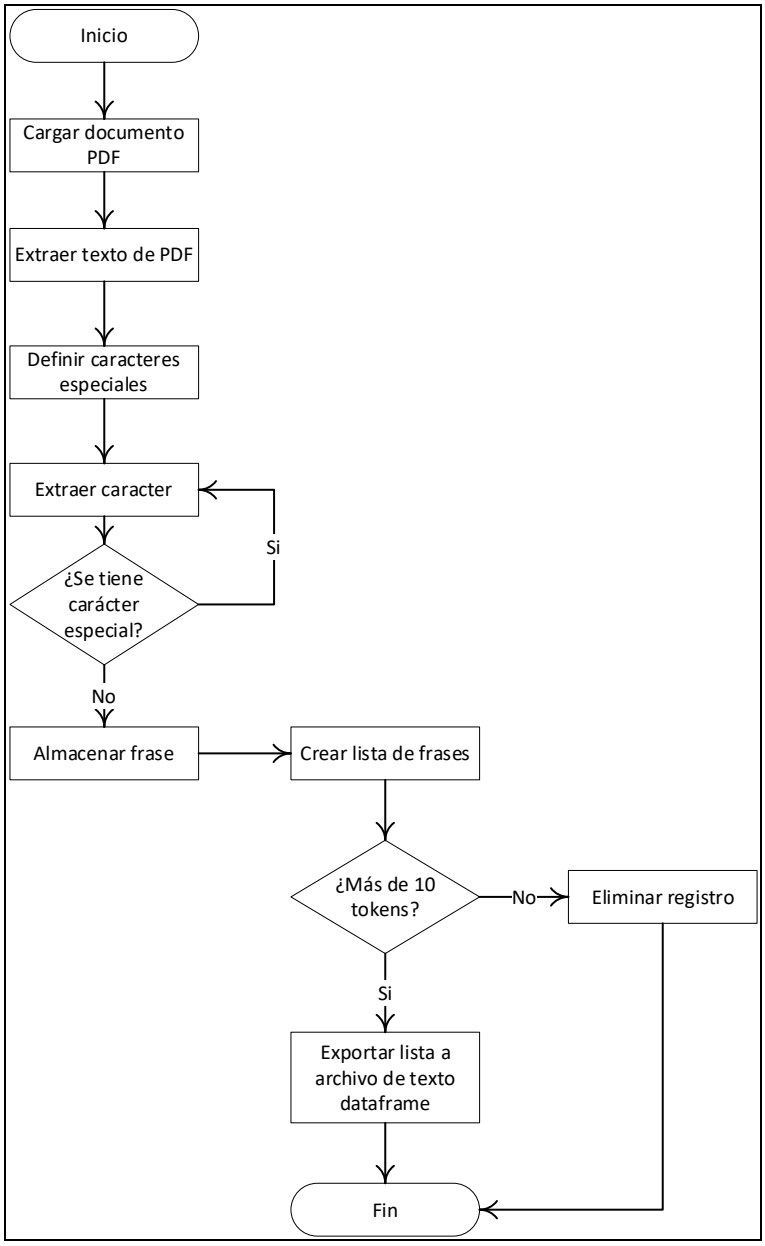
El procedimiento de extracción de texto de los planes de gobierno en PDF fue realizado de forma automática, y aunque esta forma no garantiza al 100% resultados de extracción exactos comparados a un humano, consume mucho menos tiempo. Para garantizar la máxima eficacia posible al momento de la extracción, el procedimiento se realizó ad-hoc por cada plan de gobierno en específico, de tal modo que primero se investigó el formato de cada documento por separado de forma manual y se enumeró una lista de características de cada documento que van desde signos de puntuación más usados, manera de identar los textos, cabeceras y pies de página hasta el rango de páginas a extraer.

Para la extracción de texto de los pdf, se empleó la librería fitz, luego de obtener el texto, el procedimiento ad-hoc fue realizado para la extracción de frases de todo el texto usando expresiones regulares con la librería regex. Este último procedimiento se ha graficado en un diagrama de flujo en la figura 3.6.

Input: Texto pdf proveniente de planes de gobierno o transcripción de debate (normalmente separado en párrafos)

Output: dataframe en .txt donde cada registro corresponde a una frase corta (o quasi-frase)

Figura 3.6 Extracción de frases en textos de forma automática



Fuente: Elaboración propia

Nota: El diagrama de flujo expuesto detalla El proceso carga un PDF, extrae su texto y define caracteres especiales para analizar el contenido. Va extrayendo caracter es para formar frases y crea una lista de ellas. Si la frase tiene más de 10 tokens se exporta, si no se elimina, y el proceso finaliza.

En el anexo 3 se detalla el código Python para el proceso de extracción de frases en texto.

3.5.2. Análisis exploratorio y limpieza de datos

El análisis exploratorio y limpieza de datos implica tratar la data fuente de tal forma que sea enteramente procesable. El preprocesamiento de los datos es un proceso crítico y esencial para cualquier modelo de Machine Learning. Los resultados de una investigación pueden variar en función a cuan limpia esté la data.

Los pasos seguidos para el preprocesamiento son:

- Tokenización: Consiste en dividir los datos no estructurados y el texto en lenguaje natural en fragmentos de información que pueden considerarse componentes discretos. Las presencia, es decir la ocurrencia de tokens en un documento se pueden usar directamente como un vector que representa ese documento.
- Eliminación de signos de puntuación
- Stopwords: Se denomina Stopwords al conjunto de palabras que básicamente son comúnmente usadas en cualquier lengua. El proceso consiste en identificar aquel conjunto de palabras que realmente son las relevantes a la hora de procurar identificar el requerimiento de fondo. Por lo tanto, es preciso determinar las palabras relevantes de las no relevantes.
- Stemming: Es el proceso de eliminar una parte de una palabra o reducir la misma a su palabra raíz. Esto podría no significar necesariamente que se está reduciendo una palabra a su raíz de diccionario esto para tener una idea base de cual es el mensaje de fondo.
- Lematización: La lematización consiste en encontrar el lema correspondiente de una palabra. Entiéndase por lema, a la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Se podría decir que la lematización consiste en estandarizar, desambiguar, segmentar y según se tenga como objetivo, etiquetar una palabra para darle una forma particular.

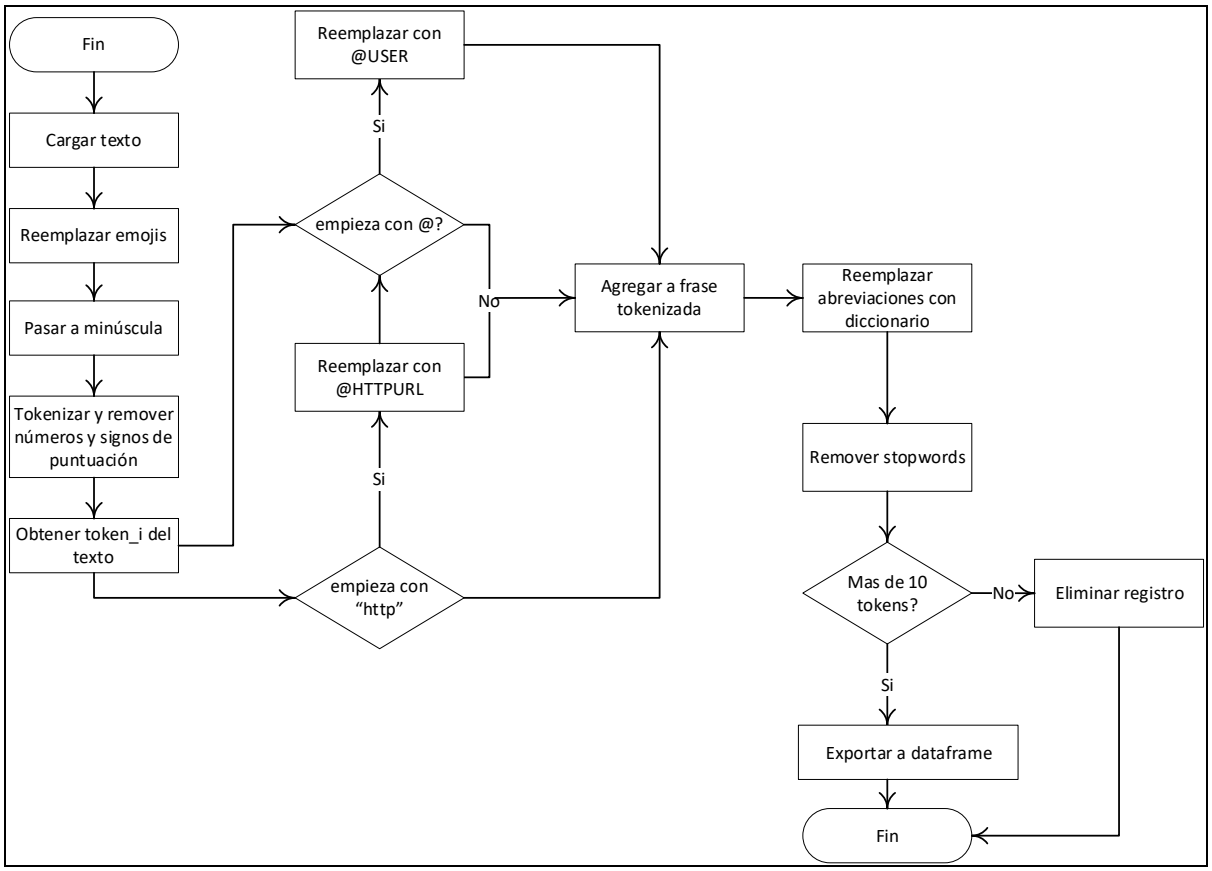
El preprocesamiento de texto engloba a la limpieza de los tweets extraídos y de los textos extraídos de los planes de gobierno y transcripciones de debates políticos. En este caso, la limpieza de los tweets y de las transcripciones de debates usan el mismo algoritmo de limpieza, mientras que las frases de los planes de gobierno son pre procesados con una variación más simple del algoritmo. Posteriormente se realiza la concatenación de los dos datasets de entrenamiento (tweets y debate) en uno solo.

La figura 3.7 muestra el diagrama de flujo del procedimiento de limpieza para tweets (y debates) mientras que la figura 3.7 muestra el diagrama de flujo del procedimiento de limpieza para los planes de gobierno.

Inputs: Tweets/Frases extraídas de debates/Frases extraídas de planes de gobierno

Outputs: Dataset limpio para ML y DL/Dataset de frases de planes de gobierno para fine-tuning de modelos de embeddings

Figura 3.7 Procedimiento de limpieza para un tweet/frase de debate político.

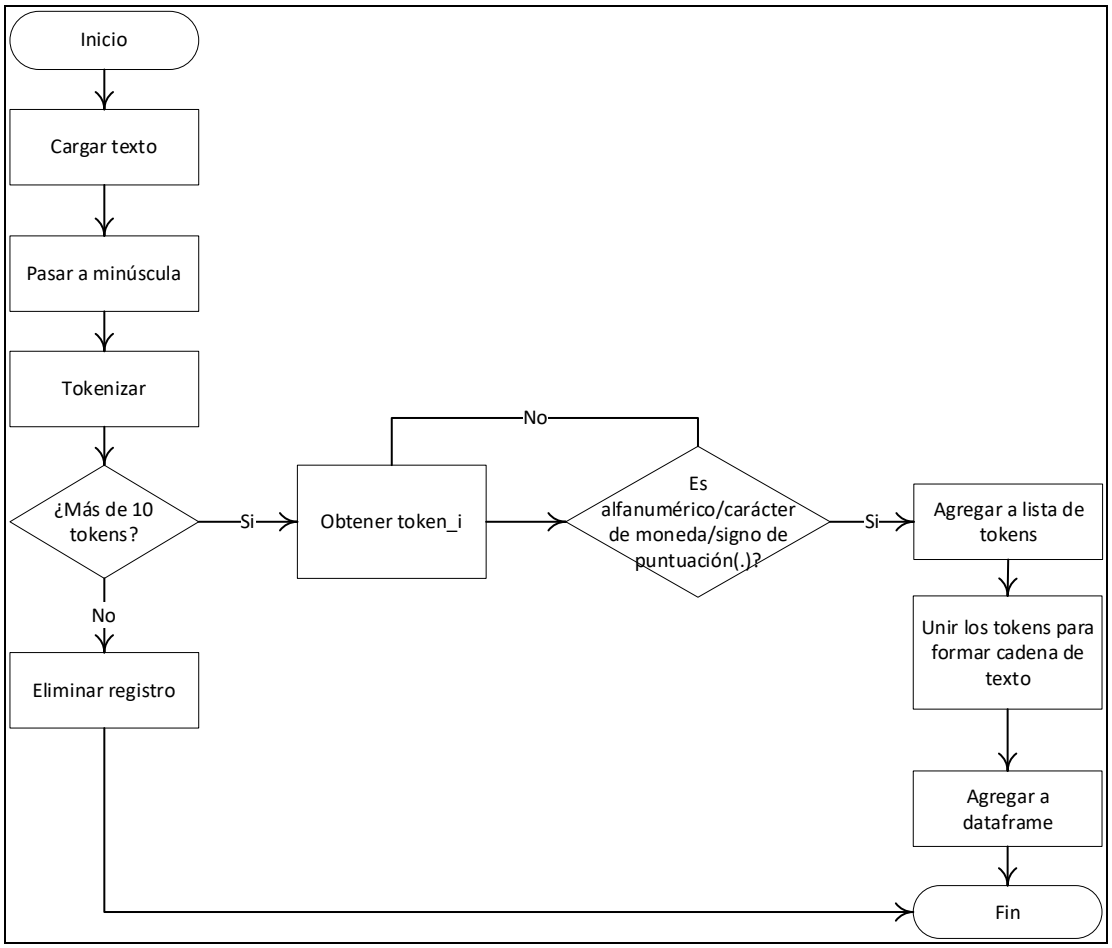


Fuente: Elaboración propia
Nota: El diagrama de flujo describe un proceso de limpieza y preprocesamiento de texto (NLP) que normaliza emojis, minúsculas y tokens, anonimiza menciones y URLs, y filtra abreviaciones y stop words. Finalmente, solo exporta a un dataframe los registros que superan los 10 tokens, eliminando los que no cumplen este requisito.

En el anexo 4 se detalla el código Python para el proceso limpieza y etiquetado de datos de tweets

En el anexo 5 se detalla el código Python para el proceso de limpieza en el texto de debates

Figura 3.8 Procedimiento de limpieza para una frase extraída de un plan de gobierno.



Fuente: Elaboración propia
Nota: El diagrama de flujo refleja el preprocesamiento de texto que filtra registros con menos de 10 tokens y normaliza el contenido a minúsculas. Valida cada token para conservar solo caracteres alfanuméricos, símbolos de moneda o puntos, reconstruyendo la cadena final para su exportación a un dataframe.

En el anexo 6 se detalla el código Python para el procesamiento y análisis en planes de gobierno

3.5.3. Vectorización de datos

La vectorización es el proceso por el cual se convierte datos de entrada de su formato sin procesar (es decir, texto) en vectores de números reales, para que sean viables procesar por los algoritmos de Machine Learning. El proceso no es nuevo y viene siendo

aplicado desde la propuesta de las primeras estructuras de programación, entre ellas vectores y matrices y punteros.

El proceso de vectorización posibilita la extracción de características distintivas del texto para que el modelo se entrene, convirtiendo el texto en vectores numéricos.

Existen muchas técnicas ya probadas para este proceso como:

- Bag of words
- TF-IDF
- TF
- Word2Vec

Para esta etapa se utilizó la librería gensim que implementa los algoritmos Word2Vec y FastText, mientras que para BoW (Bag of Words) se empleó la librería scikit-learn. Cabe resaltar que Word2Vec y FastText fueron no solo usados en los experimentos con Machine Learning, sino también para los experimentos con Deep Learning. Mientras que BoW solo fue usado para Machine Learning.

Para la implementación de este etapa, es necesario especificar que se tienen dos tipos de implementación: para Machine Learning y para Deep Learning, la razón va por la forma en como se representan los datos (vectores para ML, matrices para DL) y también el consumo de recursos (DL consume muchas más memoria que ML y por lo tanto se requiere una implementación más eficiente).

Respecto a los algoritmos de Machine Learning se hizo uso de las siguientes librerías:

SVM.py: La clase principal para algoritmos de Machine Learning que implementa una pipeline de sklearn para llevar a cabo el proceso de vectorización y entrenamiento de forma automática. Las demás implementaciones de código se conectan a dicha pipeline:

Vectorizer.py: Implementa la clase SentenceEmbeddingVectorizer cuya función es vectorizar con Word2Vec y FastText la data de entrenamiento

EmbeddingLoader.py: Se encarga del manejo de carga de los modelos de embedding preentrenados utilizados para los experimentos.

ColumnExtractor.py: Se encarga de extraer la columna que contiene las frases en el dataset de entrenamiento para posteriormente ser vectorizadas.

Respecto a los algoritmos de Deep Learning se hizo uso de las siguientes librerías:

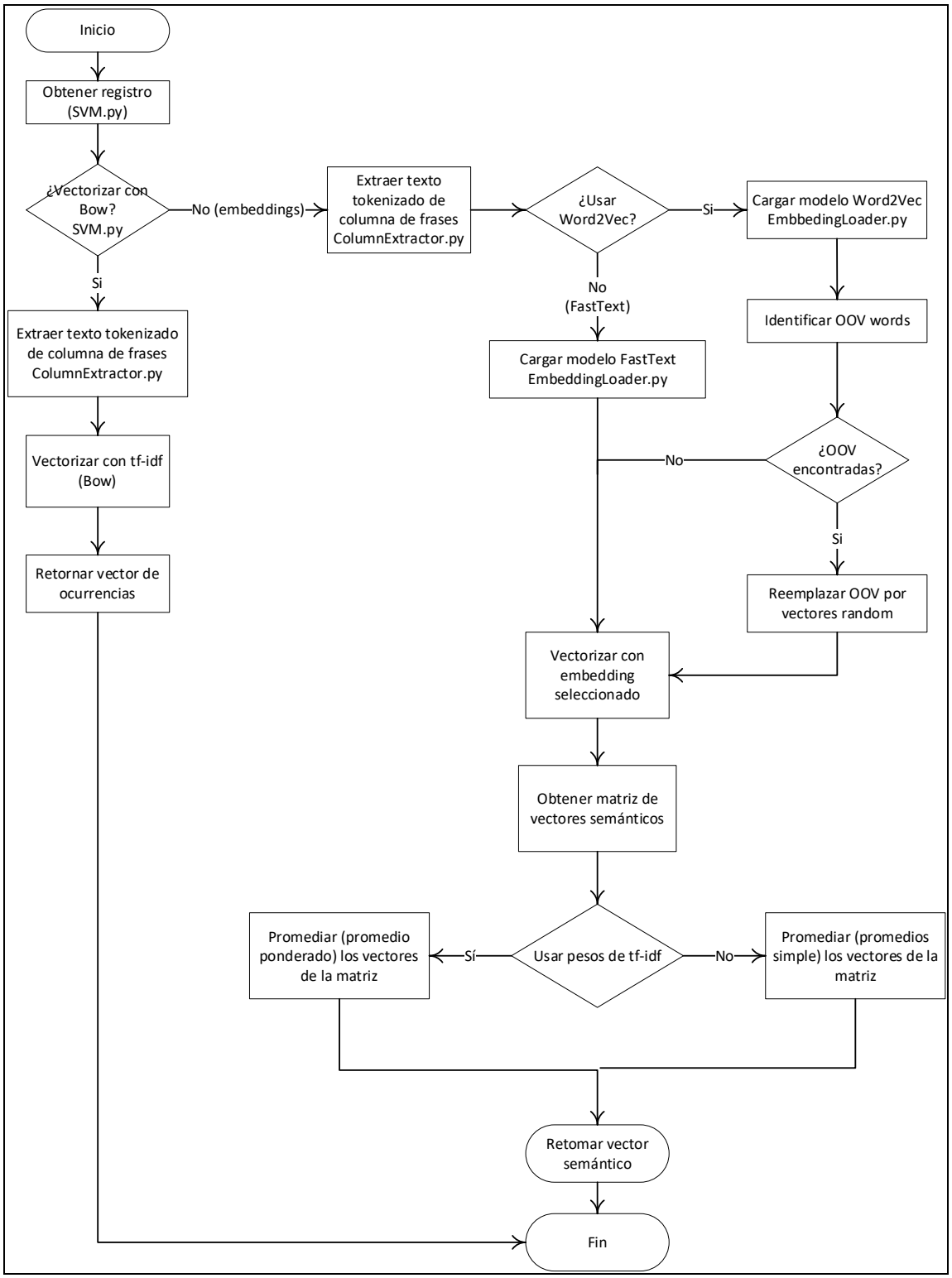
DataGenerator.py: Una clase que implementa la función Generator de la API keras, su función es generar batches de frases vectorizadas dado un modelo embedding preentrenado, retorna un batch de matrices, donde cada matriz corresponde a una frase vectorizada. Es self-contained, lo cual significa que puede ser usada para cualquier algoritmo de DL sin tener que depender de un pipeline en específico.

Inputs: Dataset de tweets y debates anotados y limpios

Outputs: Matriz de vectores/Batch de Matrices de vectores semánticos (cada matriz corresponde a un registro/frase)

Los diagramas de flujo para cada tipo de vectorización de Machine Learning se detalla en la figura 3.9 y los de Deep Learning se detalla en la figura 3.10.

Figura 3.9 Proceso de vectorización para experimentos con Machine Learning.

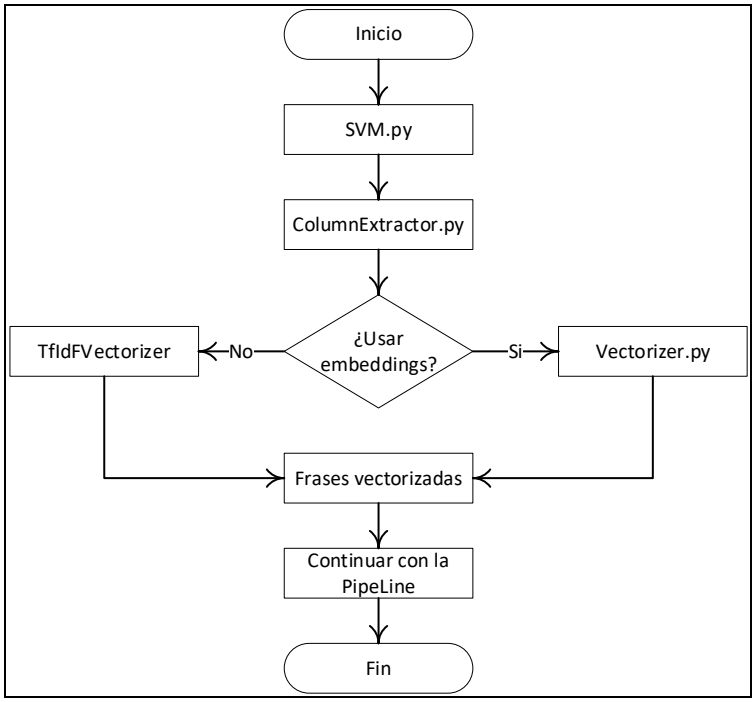


Fuente: Elaboración propia

Nota: El diagrama de flujo describe un sistema de vectorización de texto que permite elegir entre Bag of Words (TF-IDF) o embeddings (Word2Vec/FastText). En el caso de los embeddings, gestiona palabras fuera del vocabulario (OOV) y genera un vector semántico final mediante promedios simples o ponderados por pesos TF-IDF.

En el anexo 7 se detalla el código Python para el proceso de vectorización para experientos con ML

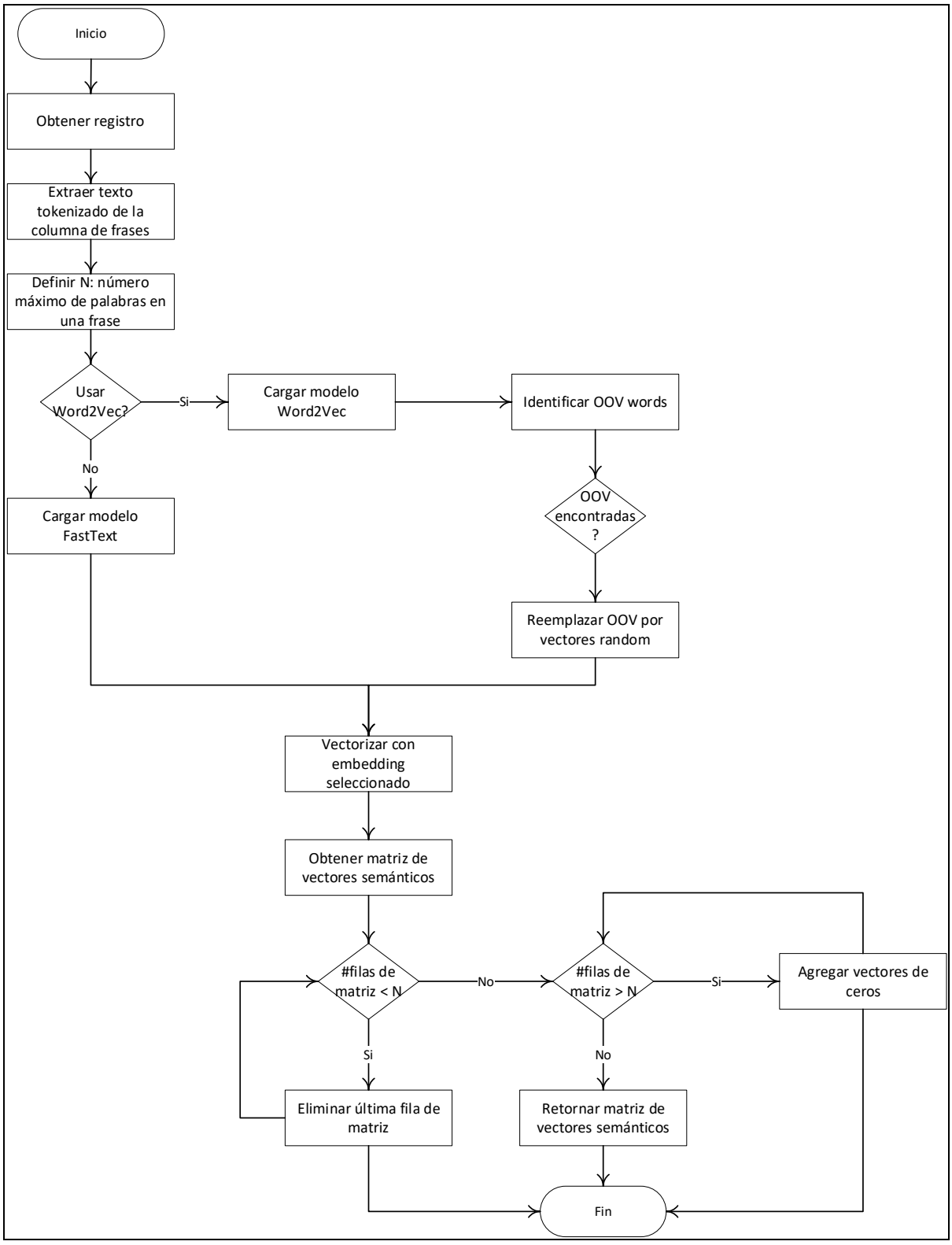
Figura 3.10 Flujo del pipeline programado para la vectorización de datos en ML.



Fuente: Elaboración propia
Nota: El diagrama de flujo ilustra un pipeline de procesamiento que extrae columnas de datos para convertirlas en vectores numéricos. Dependiendo de la decisión, utiliza TF-IDF (método estadístico) o embeddings (método semántico) para generar las frases vectorizadas que continuarán en el flujo de ejecución.

En el anexo 8 se detalla el código Python para el proceso

Figura 3.11 Proceso de vectorización para experimentos con Deep Learning.



Fuente: Elaboración propia

Nota: El diagrama de flujo refleja el proceso de generación de una matriz de vectores semánticos mediante Word2Vec o FastText, gestionando palabras desconocidas (OOV) con vectores aleatorios. Finalmente, aplica un truncamiento o padding (relleno con ceros) para asegurar que cada frase tenga exactamente una longitud N de palabras antes de retornar la matriz.

En el anexo 9 se detalla el código Python para el proceso de vectorización para experimentos en Deep Learning.

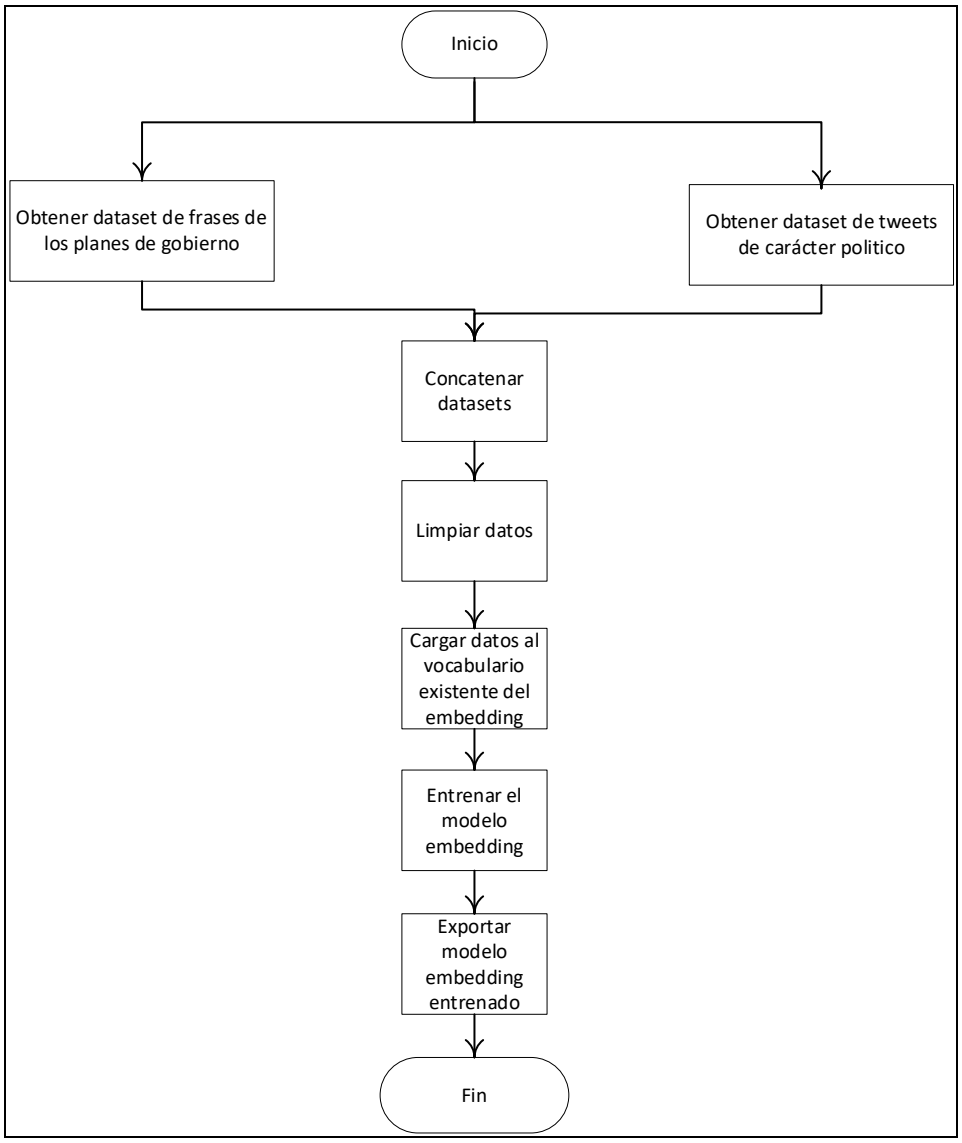
Finalmente, en esta etapa, respect al proceso Fine-tuning a los modelos de vectorización usando datos del dominio politico se consideró lo siguiente:

Inputs: Modelo de embedding preentrenado/Frases extraídas de partidos políticos/tweets extraídos de dominio poílitco

Outputs: Modelo embedding entrenado con datos de dominio político

En la figura 3.12 se detalla el diagrama de flujo de esta etapa.

Figura 3.12 Proceso de carga de datos y fine-tuning al modelo de words embedding.



Fuente: Elaboración propia
Nota: El diagrama de flujo describe un proceso de fine-tuning de embeddings, donde se concatenan y limpian datasets de planes de gobierno y tweets políticos para actualizar un vocabulario existente. Posteriormente, se entrena el modelo de embedding con estos datos específicos del dominio y se exporta el modelo final optimizado.

En el anexo 10 se detalla el código Python para el proceso de creación del corpus de entrenamiento

En el anexo 11 se detalla el código Python para el proceso fine tuning

3.5.4. Aplicación de transformadores y clasificación

Los transformadores (Transformer) es una técnica novedosa que buscó en su primera versión realizar traducciones entre lenguajes naturales combinando el proceso con redes neuronales. La estructura de un Transformer contempla un Encoder y un Decoder. El encoder se utiliza para analizar el contexto de la secuencia de entrada y el decoder es el encargado de generar la secuencia del output a partir de este contexto. El proceso realizado por el encoder y decoder es complementado por un proceso intermedio llamado embedding; este proceso posibilita tener una representación vectorial de las palabras. El objetivo es conseguir una representación numérica de las palabras. La idea con el embedding es que gracias a su clasificación numérica vectorial y su significado semántico representarlas en un plano y determinar su proximidad.

3.5.5. Afinamiento

El afinamiento dentro del framework propuesto consiste en la aplicación de diversos modelos pre entrenados que potencialmente pueden servir en la aplicación. Dadas las características de la investigación y su metodología es preciso combinar y/o evaluar el algoritmo pertinente. Para la investigación se consideraron los siguientes algoritmos:

- Word embeddings
- Support vector machine
- Redes neuronales convolucionales (CNN)
- CNN – Simple
- Redes neuronales recurrentes (RNN)

3.5.6. Evaluación y selección de modelo

Para efectos de evaluar el rendimiento de los modelos usados, se recurre a métricas de precisión, dentro de las que se destaca: Recall, F1, Accuracy y la matriz de confusión.

Para la investigación se utilizó la métrica F1. El valor F1 se utiliza para combinar las medidas de precisión y recall en un sólo valor. Con esta métrica se asume como componentes relevantes de evaluación la precisión y la exhaustividad.

Los modelos evaluados con esta métrica en la investigación fueron:

- Support Vector Machine
- Redes neuronales convolucionales (CNN)
- Redes neuronales convoluacionales simples
- Redes neuronales recurrentes (RNN)

Para la implementación de estos modelos se utilizó la librería scikit-learn para Machine Learning y tensorflow para Deep Learning. Esta etapa es la que representa a la fase de experimentación del proyecto, con el objetivo de facilitar los diferentes experimentos llevados a cabo con todos los modelos, se diseñó e implementó un conjunto de módulos que se encargan del proceso de vectorización, entrenamiento, fine-tuning y evaluación.

A continuación se detallan las librerías según tipo de experimentación (Machine Learning o Deep Learning):

Machine Learning:

SVM.py: La clase principal para algoritmos de Machine Learning que implementa una pipeline de sklearn para llevar a cabo el proceso de vectorización y entrenamiento de forma automática. Las demás implementaciones de código se conectan a dicha pipeline:

Vectorizer.py: Implementa la clase SentenceEmbeddingVectorizer cuya función es vectorizar con Word2Vec y FastText la data de entrenamiento

EmbeddingLoader.py: Se encarga del manejo de carga de los modelos de embedding preentrenados utilizados para los experimentos.

ColumnExtractor.py: Se encarga de extraer la columna que contiene las frases en el dataset de entrenamiento para posteriormente ser vectorizadas.

Deep Learning:

CNN.py: La clase principal para los algoritmos de Deep Learning, se encarga del entrenamiento de diversos modelos de Deep Learning y también implementa el proceso de fine-tuning de hiperparámetros de manera automatizada. Esta clase recibe a "hyper-modelos" los cuales poseen un conjunto de parámetros que luego la clase principal internamente se encarga de ordenar para realizar el fine-tuning usando la librería KerasTuner.

CNN_hypermodel.py: Clase que implementa al modelo CNN y a todo el conjunto de hiperparámetros que se van a usar para el proceso de fine-tuning de hiperparámetros.

GRU_hypermodel.py: Clase que implementa al modelo GRU y a todo el conjunto de hiperparámetros que se van a usar para el proceso de fine-tuning de hiperparámetros.

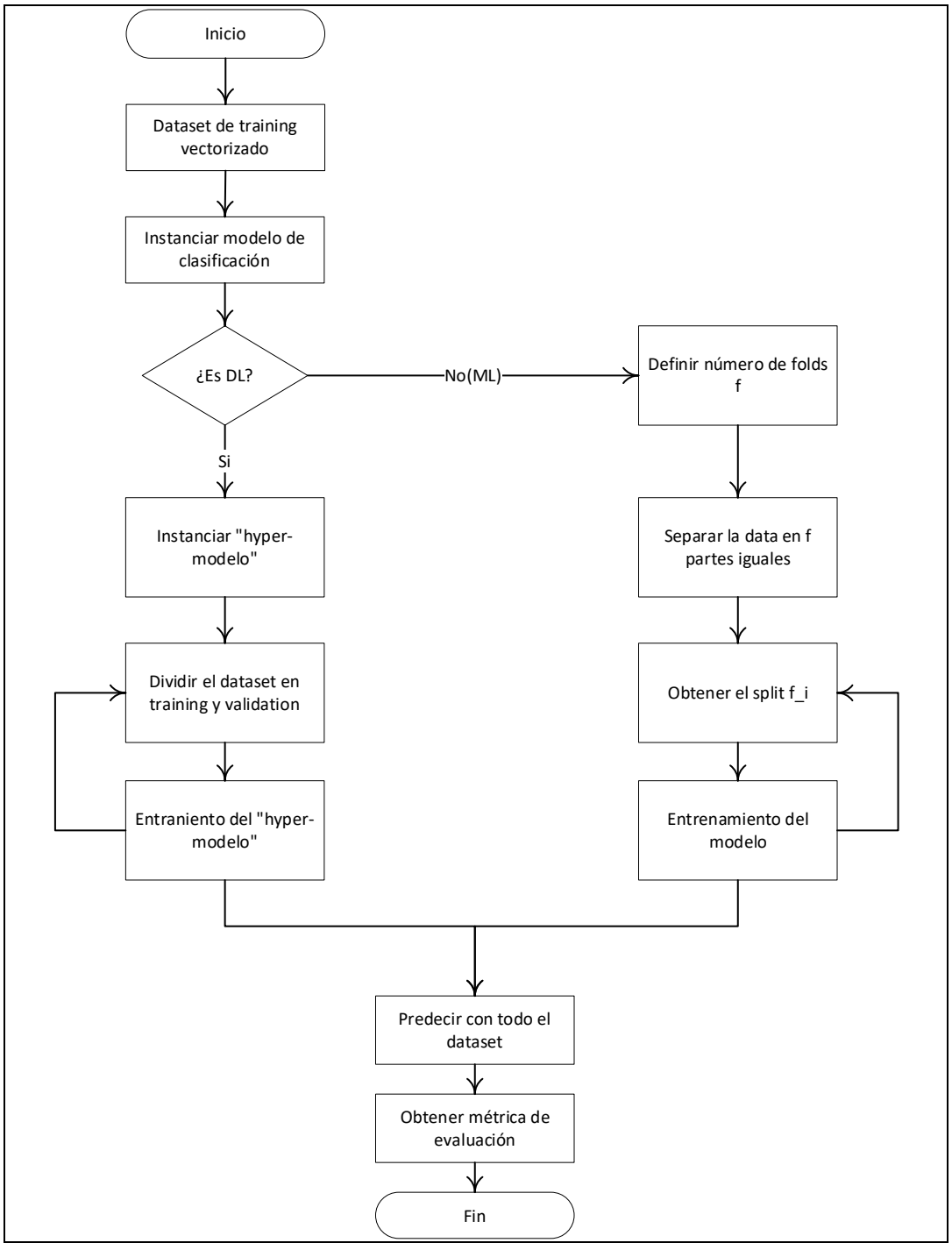
Con los módulos descritos es posible realizar los experimentos de forma automática, los jupyter notebooks en Python usados para la experimentación (y fine-tuning) para los modelos CNN y GRU.

La etapa de evaluación se hace en conjunto con la etapa de clasificación y entrenamiento, las métricas utilizadas en la evaluación para medir la eficacia de los modelos fueron el F1 Score (micro y macro) y los valores productos de la los function para el caso de los modelos de Deep Learning, estos valores los sirven para elegir el número óptimo de épocas de entrenamiento y los hiperparámetros más óptimos.

El diagrama de flujo del proceso de entrenamiento y evaluación de los modelos se encuentra descrito en la figura 3.13.

Para fines de replicabilidad y reproducibilidad de los experimentos realizados en este proyecto.

Figura 3.13 Proceso de entrenamiento, fine-tuning y evaluación de los modelos.



Fuente: Elaboración propia

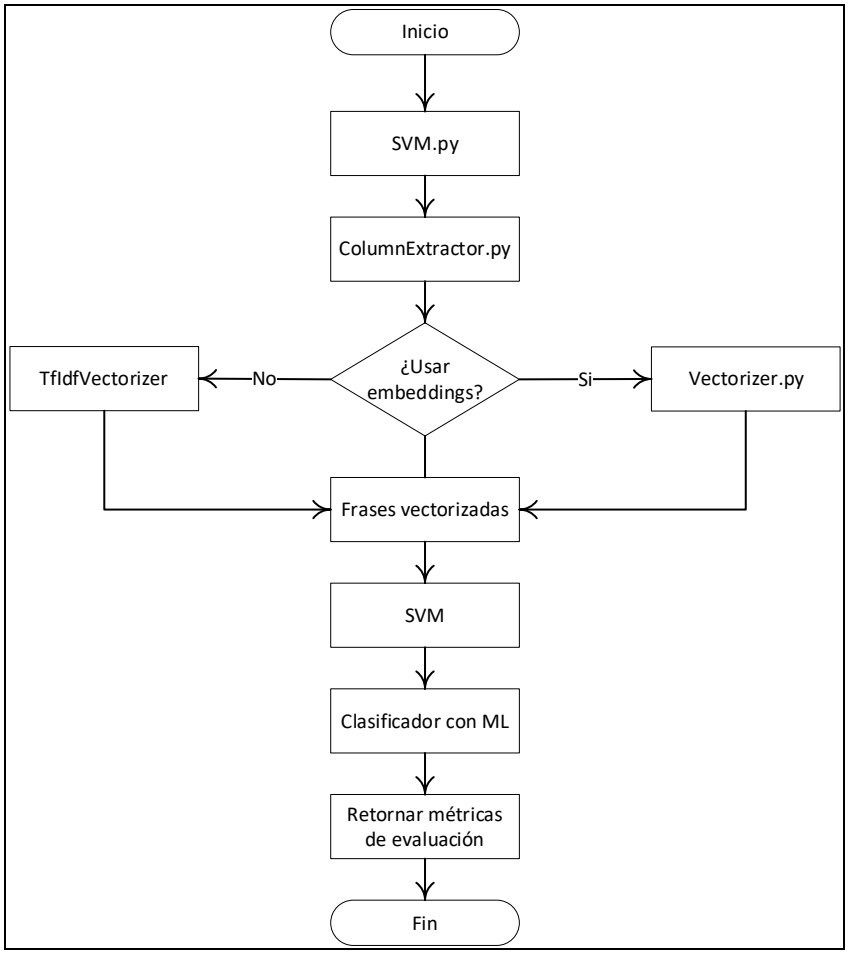
Nota: El diagrama de flujo refleja el proceso de entrenamiento de modelos de clasificación diferenciando entre Deep Learning (uso de "hyper-modelos" con validación simple) y Machine Learning tradicional (aplicando K-Fold Cross Validation). Al finalizar el

entrenamiento iterativo, realiza predicciones sobre el dataset completo para calcular y obtener las métricas de evaluación finales.

En el anexo 12 se detalla el código Python para el proceso experimental usando CNN

En el anexo 13 se detalla el código Python para el proceso experimental usando GRU

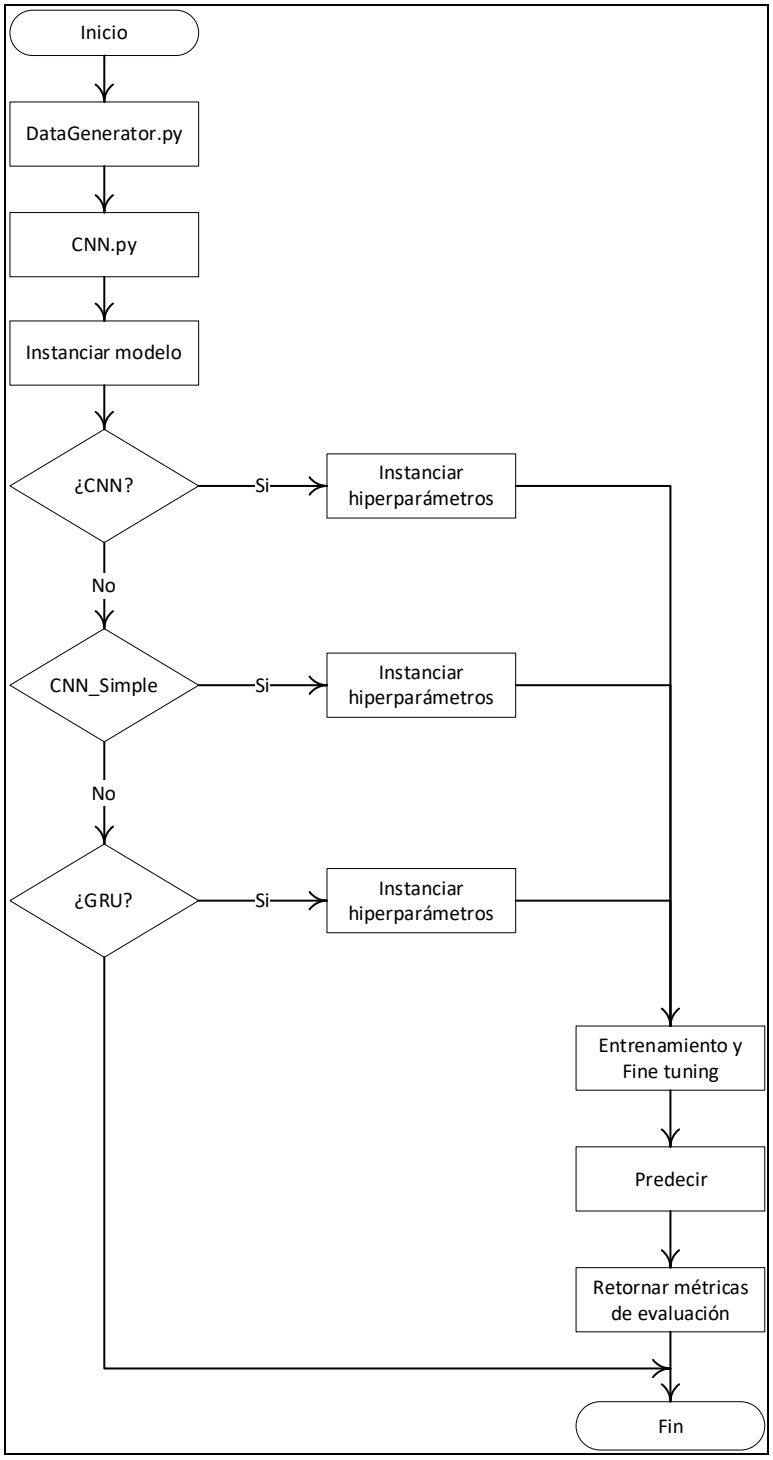
Figura 3.14 Flujo del pipeline programado para clasificación usando técnicas de Machine Learining.



Fuente: Elaboración propia

En el anexo 14 se detalla el código Python para el proceso de clasificación de tópicos políticos usando Machine Learning.

Figura 3.15 Flujo del modulo programado para la clasificación usando técnicas de Deep Learning.



Fuente: Elaboración propia
Nota: El diagrama de flujo describe la selección y entrenamiento de arquitecturas de Deep Learning (CNN, CNN Simple o GRU), donde se instancian hiperparámetros específicos

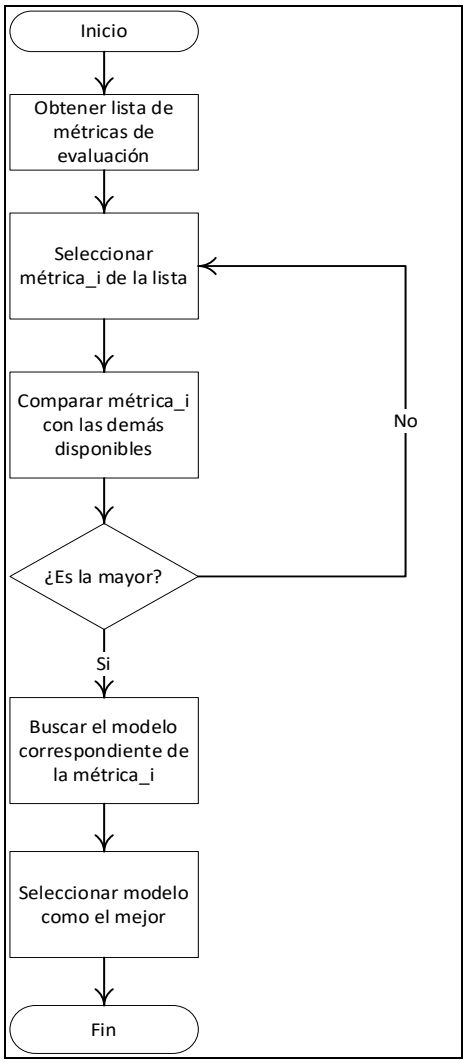
según el modelo elegido. Tras el entrenamiento y fine-tuning, el sistema genera predicciones y retorna las métricas de evaluación para validar el rendimiento.

En el anexo 14 se detalla el código Python para el proceso de clasificación de tópicos políticos usando Deep Learning.

Respecto al proceso de la selección de mejor modelo, luego del proceso de entrenamiento y evaluación, se lleva a cabo la selección del mejor modelo tomando en cuenta las métricas de evaluación resultantes, las razones de selección, resultados/scores de evaluación.

Un diagrama de flujo sobre esta etapa está disponible en la figura 3.16.

Figura 3.16 Proceso de selección del mejor modelo.



Fuente: Elaboración propia
Nota: El diagrama de flujo refleja el proceso de comparación iterativa de métricas de evaluación para identificar el valor máximo dentro de una lista. Una vez hallada la métrica más alta, el flujo identifica y selecciona el modelo asociado como el mejor de la comparativa para finalizar el ciclo.

Finalmente se detalla el conjunto de librerías utilizadas en el proceso de experimentación:

twint: Librería para extracción y scraping de tweets de forma automática.

PyMuPDF (fitz): Librería para la extracción de textos en formato PDF, se utilizó en la etapa de extracción de textos específicamente para extraer el texto de los planes de gobierno y las transcripciones de debates políticos de forma automatizada.

pandas: Manejo de dataframes, se usó en todas las etapas del proyecto.

matplotlib: Librería para realizar gráficos (barras, pie, scattering, etc), usada para el análisis exploratorio.

numpy: Librería de alto rendimiento, especializada en operaciones númericas con arrays, se usó en todas las etapas del proyecto.

scikit-learn: Librería que contiene algoritmos de ML, así como algoritmos para preprocesamiento de textos, se usó para toda la etapa de experimentación con algoritmos de Machine Learning.

gensim: Librería especializada en algoritmo de topic-modelling, contiene implementaciones de modelos de vectorización embeddings como word2vec y fasttext, se usó para la etapa de vectorización y fine-tuning de fasttext.

tensorflow: Librería que contiene implementaciones de algoritmos de DL, así como algoritmos para preprocesamiento de textos, se usó para toda la etapa de experimentación con algoritmos de Deep Learning, específicamente, se empleó la API de alto nivel llamada Keras que es parte de tensorflow.

KerasTuner: Es un módulo independiente de la API Keras para realizar el proceso de fine-tuning de hieperprámetros, contiene diversos algoritmos de búsqueda y optimización de parámetros. Esta librería fue empleada para el proceso de fine-tuning de hiperparámetros a los modelos basados en Deep Learning, es parte del módulo automatizado desarrollado para este proyecto.

CAPÍTULO 4 RESULTADOS Y DISCUSIÓN

4.1. Extracción y procesamiento de los datos

Los resultados del preprocesamiento evidencian que las técnicas de limpieza, normalización y tokenización permitieron reducir significativamente el ruido presente en los datos textuales provenientes de redes sociales, planes de gobierno y transcripciones del debate presidencial. Este proceso facilitó la construcción de representaciones vectoriales consistentes, necesarias para el posterior entrenamiento de los modelos de Machine Learning.

Asimismo, la extracción de características permitió identificar términos y patrones discursivos recurrentes asociados a los principales tópicos políticos analizados, lo que confirma la pertinencia del enfoque basado en Procesamiento de Lenguaje Natural para el tratamiento de información política no estructurada.

4.1.1. Debates presidenciales

Se extrajo transcripciones de los debates presidenciales oficiales organizados por el JNE (Jurado Nacional de Elecciones) de Perú en 3 fechas consecutivas donde el primero de estos se llevó a cabo el lunes 20 de marzo de 2021. Asimismo, también se recolectó las transcripciones del debate presidencial llevado a cabo durante la segunda vuelta electoral entre los candidatos Keiko Fujimori y Pedro Castillo. Estos debates presidenciales se caracterizan por tener tópicos de discusión y cada candidato puede hablar por un intervalo determinado de tiempo. Las transcripciones se hicieron en forma de párrafos, donde cada

uno de estos párrafos pertenece a un tópico. La tabla 4.1 muestra la cantidad de párrafos por cada tópico.

Tabla 4.1: Organización de tópicos en debates presidenciales

Tópico	Cantidad de párrafos
Seguridad	128
Pandemia	127
Corrupción	50
Educación	48
Economía	12

Fuente: Elaboración propia
Nota: La tabla muestra la distribución de párrafos por tópico. Los temas más predominantes son Seguridad (128) y Pandemia (127), lo que indica que concentran la mayor parte del contenido. Corrupción (50) y Educación (48) tienen una presencia moderada, mientras que Economía (12) es el tema menos abordado.

Los tópicos de estos debates fueron usados como punto de partida para definir los tópicos de clasificación, ya que, se entiende, que no existen fuentes de datos con textos políticos etiquetados por categorías para el contexto peruano.

Con el objetivo de adecuar las transcripciones al formato del Manifesto Project, se divide cada párrafo en unidades más pequeñas, similar a las pseudo-oraciones o "quasi-sentences" usadas en el Manifesto Project (Manifesto, 2021). Para esto, se creó aproximaciones a una pseudo-oración diviendo cada párrafo en oraciones más cortas separadas por un punto o un punto y coma. En la tabla 4.2 se muestran a detalle los tópicos por párrafo y pseudo-oraciones.

Tabla 4.2: Número de pseudo oraciones en transcripciones

Tópico	Párrafos	POs
Seguridad	128	633
Pandemia	127	624
Corrupción	50	251
Educación	48	217
Economía	12	14

Fuente: Elaboración propia
Nota: La tabla muestra la relación entre tópicos, cantidad de párrafos y POs. Los temas Seguridad (128 párrafos, 633 POs) y Pandemia (127 párrafos, 624 POs) concentran la mayor cantidad de contenido y opiniones. Corrupción y Educación presentan una presencia intermedia, mientras que Economía tiene la menor participación (12 párrafos y 14 POs), lo que indica poca representación en el conjunto de datos

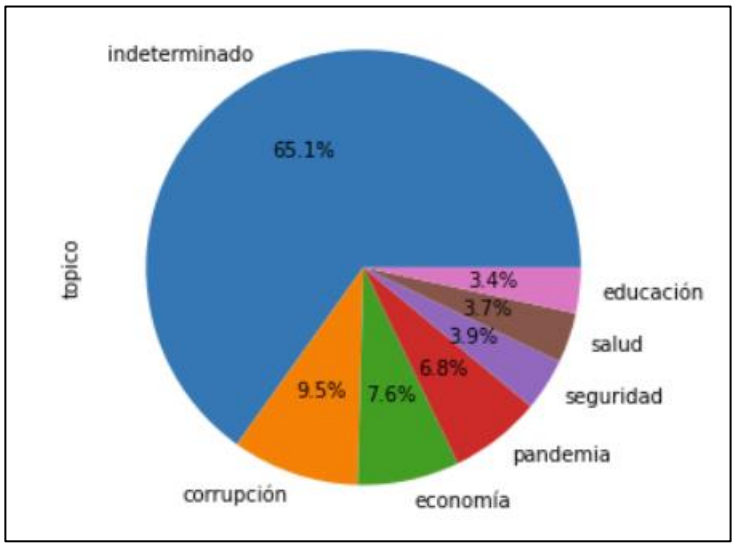
4.1.2. Data procesada de Twitter

Se extrajeron un total de 5689 tweets correspondientes a la campaña política de elecciones 2021 en Perú, el periodo de extracción fue de del 1 de enero al 10 de abril de 2021 y 12 de abril al 5 de junio de 2021 sumando un total de 153 días. No se consideraron los tweets que fueron publicados durante la fecha misma de elecciones ya sea para primera o segunda vuelta, esa es la razón de porque hay brechas entre los periodos de extracción. Solo se extrajeron tweets de las cuentas oficiales de los candidatos a la presidencia, no se consideraron retweets ni respuestas/comentarios de los usuarios hacia estos tweets. De esta forma se garantiza que los textos sean objetivos y representen verdaderamente al candidato que los escribe.

Posteriormente a la extracción, los tweets fueron etiquetados por 3 anotadores, estudiantes de ciencias políticas siguiendo los lineamientos de tópicos encontrados en los debates presidenciales con la adición del tópico “salud”. Asimismo, se agregó la etiqueta “indeterminado” que los anotadores podían utilizar cuando encuentren un tweet con un significado ambiguo o que el tópico al que este hace mención no se encuentre dentro de los lineamientos.

Los tópicos utilizados son: Seguridad, Pandemia, Corrupción, Educación, Economía, Salud e Indeterminado. La distribución de tweets con respecto a su categoría se observa en la figura 4.1.

Figura 4.1 Distribución de tópicos en porcentaje.



Fuente: Elaboración propia
Nota: Este gráfico de pastel muestra la distribución de temas en un dataset, donde el 65.1% de los registros son "indeterminados". De las categorías identificadas, la corrupción (9.5%) y la economía (7.6%) son los tópicos predominantes, seguidos por pandemia, seguridad, salud y educación.

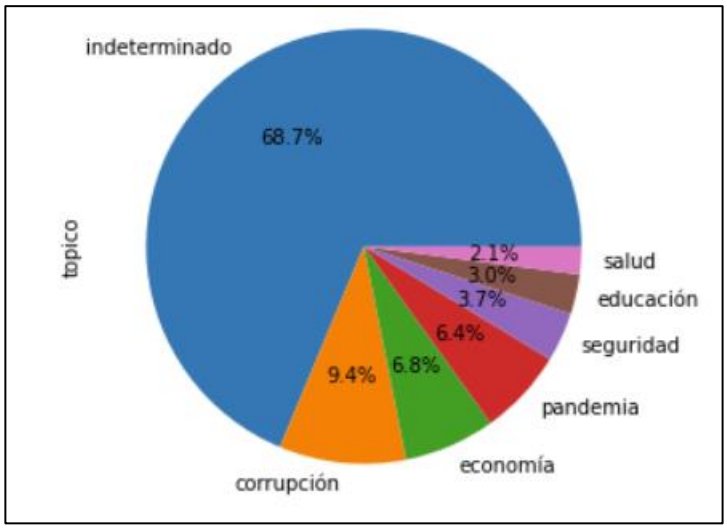
En la figura 4.1 que más de la mitad de los tweets corresponden a la categoría “indeterminado”, esto puede ser debido a que las categorías con las que se están trabajando son muy específicas o que gran parte de los candidatos usen Twitter para otros fines de campaña política que no se relacionan directamente con las categorías en estudio. La distribución también es consistente con el estudio de (Lopez Lopez et al., 2018) sobre las elecciones presidenciales anteriores, donde los autores encontraron que alrededor del 50% de los tweets publicados por los dos candidatos a la segunda vuelta de esa fecha (PPK y Keiko Fujimori) eran referentes a la promoción de su campaña política, mientras que los tweets referidos a tópicos específicos, como los usados en esta investigación, aparecían en un menor porcentaje.

El tópico de pandemia es especial debido a que las elecciones de 2021 también estuvieron enfocadas en temas sobre cómo enfrentar a la COVID-19, este tópico probablemente no se vuelva recurrente o deje de existir en próximas elecciones.

A continuación, se muestra la misma distribución de tweets por tópicos por orientación política, es decir por izquierda y derecha.

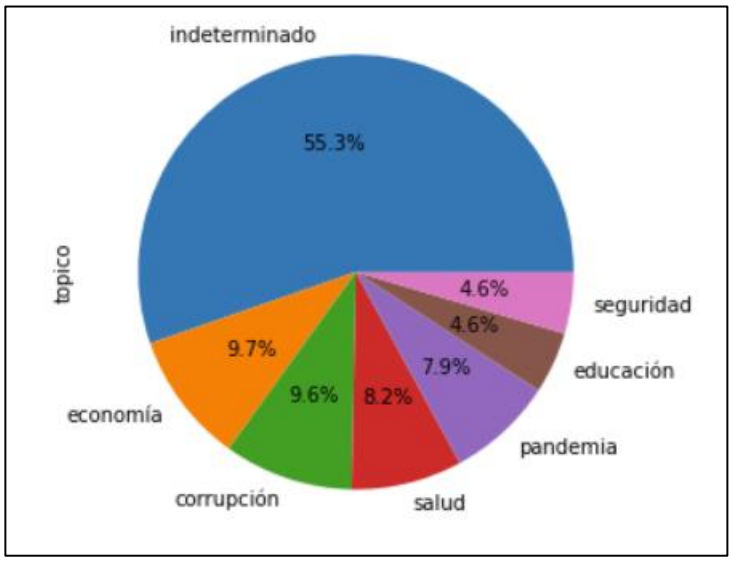
Tomando en consideración la línea política, según lo expuesto en las figuras 4.2 y 4.3, existe un mayor porcentaje de tweets “indeterminados” en la derecha que, en la izquierda, pero la diferencia es minima.

Figura 4.2 Distribución de tweets por tópicos – Partidos de “La derecha”



Fuente: Elaboración propia
Nota: Este gráfico revela que casi el 69% de los tópicos son indeterminados, lo que indica un fuerte desequilibrio o falta de clasificación. Entre las categorías identificadas, la corrupción (9.4%) y la economía (6.8%) lideran el volumen, mientras que la salud es el tema con menor representación (2.1%).

Figura 4.3 Distribución de tweets por tópicos – Paridos de “La izquierda”



Fuente: Elaboración propia
Nota: El gráfico muestra una distribución de tópicos donde el 55.3% permanece como indeterminado, evidenciando una ligera mejora en la clasificación respecto a versiones anteriores. De los temas identificados, la economía (9.7%) y la corrupción (9.6%) son los más relevantes, seguidos de cerca por salud y pandemia.

Los candidatos de derecha han publicado más que los de izquierda durante todo el periodo de campaña. Un total 4160 tweets contra 1529 de los candidatos de izquierda. En la tabla 4.3, se observa que el candidato Rafael López Aliaga fue el que más publicó en twitter durante el periodo de campaña, mientras que, en la izquierda, el candidato Marco Arana fue el que más tweets publicó.

Debido a que los tweets extraídos no cuentan con etiqueta alguna que los categorice acorde a las categorías encontradas en los debates, fue necesario etiquetar estos datos manualmente. Para este procedimiento, se requirió de 3 anotadores, para esto se tomó los servicios de estudiantes universitarios de ciencias políticas a los cuáles se les proporcionó el mismo dataset de tweets bajo una misma interfaz digital que les permitía seleccionar la categoría sin la necesidad de escribirla, de modo que así se puede prevenir errores como tweets con categorías nulas o en blanco, así como inconsistencias en la forma de tipear una categoría. Asimismo, estos anotadores fueron instruidos previamente sobre los criterios para realizar el proceso de categorización.

En adición a las categorías propuestas en la investigación, también se agregó una nueva con el nombre de “indeterminado” la cual los anotadores podrían utilizar cuando se

encuentren con un tweet que (1) no tenga relevancia con ninguna de las categorías estudiadas, (2) sea muy difícil de interpretar debido al contexto del mismo mensaje o (3) se considere que el tweet corresponda a más de una categoría. La etiqueta “indeterminado” fue la más recurrente con respecto a las demás categorías.

Para validar las anotaciones realizadas, se calcula los diversos scores de kappa (Hallgren, K. 2012) dando como resultado de acuerdo a lo registrado por los 3 anotadores un Light’s kappa de 0.37, el que representa el promedio de todos los kappa resultantes entre las parejas de anotadores. Por cada pareja de anotadores se calculó el Byrt’s kappa. El acuerdo encontrado entre los 3 anotadores corresponde a un acuerdo justo según (Landis y Koch, 1977).

En las figuras 4.4 y 4.5 se exponen las distribuciones de tópicos por candidato

Tabla 4.3: Tweets distribuidos por posición política, candidato y tópico

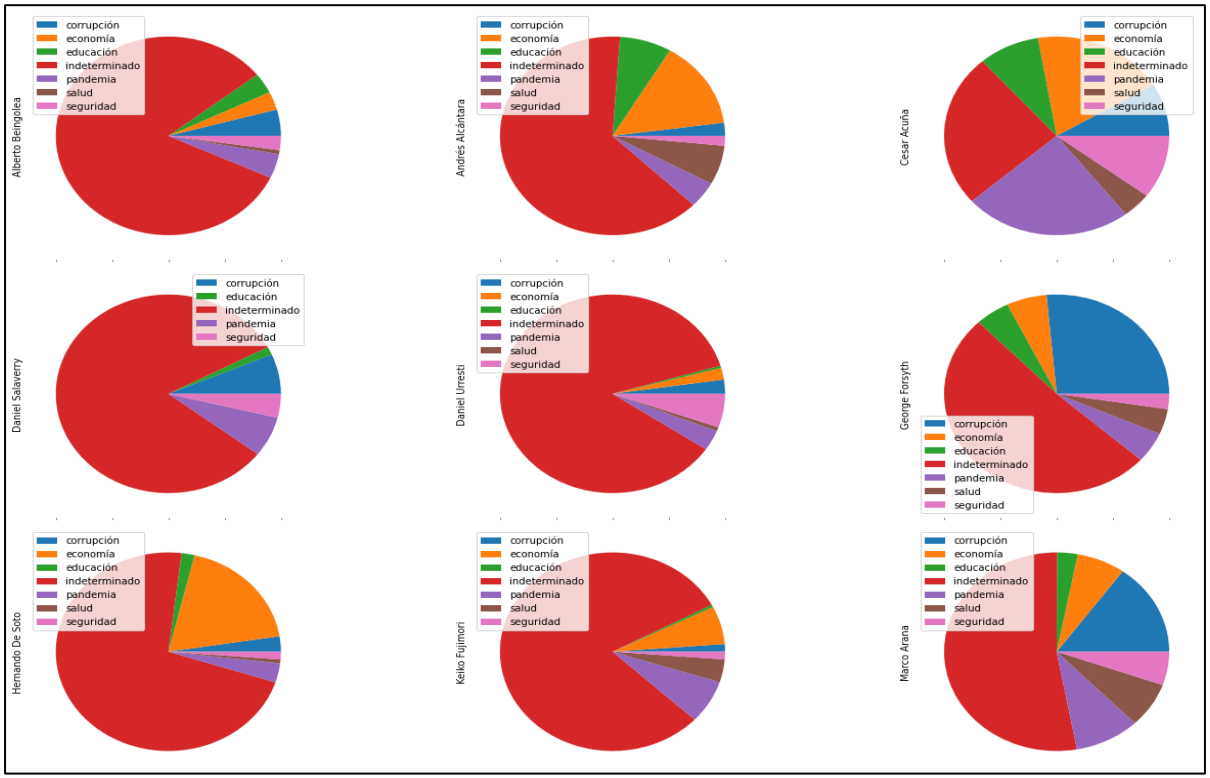
Posición	Candidato	Tópico							
		corrupción	economía	educación	indeterminado	pandemia	salud	seguridad	Total
Derecha	Alberto Beingolea	26	18	20	501	24	4	14	607
	Cesar Acuña	41	91	42	119	115	20	49	477
	Daniel Salaverry	5	0	1	63	5	0	3	77
	Daniel Urresti	18	15	3	675	26	5	43	785
	George Forsyth	32	7	6	62	6	5	3	121
	Hernando De Soto	4	30	3	115	5	1	2	160
	Keiko Fujimori	3	15	1	194	17	9	3	242
	Rafael Lopez Aliaga	202	86	36	889	56	32	31	1332
	Rafael Santos	57	11	7	182	7	3	3	270
	Yonhy Lescano	4	9	4	58	6	7	1	89
Izquierda	Andrés Alcántara	4	27	14	120	8	12	3	188
	Marco Arana	127	59	25	448	79	62	46	846
	Ollanta Humala	2	15	11	50	5	18	1	102
	Pedro Castillo	13	14	3	86	8	8	5	137
	Verónika Mendoza	1	34	17	142	21	26	15	256
									5689

Fuente: Elaboración propia

Nota: La tabla muestra la distribución de tweets por candidato, posición política y tópico. En general, el tema “indeterminado” concentra la mayor cantidad de tweets en casi todos los candidatos, lo que indica que muchos mensajes no se pueden clasificar claramente en un tópico específico. En la derecha, el candidato con más tweets es Rafael López Aliaga (1332), seguido por Daniel

Urresti (785) y Alberto Beingolea (607). En este grupo destacan temas como corrupción, economía y pandemia, aunque nuevamente predomina el tópico indeterminado. En la izquierda, el candidato con mayor cantidad de tweets es Marco Arana (846), seguido por Verónika Mendoza (256) y Andrés Alcántara (188). En estos candidatos también predomina el tópico indeterminado, pero se observa mayor presencia relativa de corrupción, pandemia y salud en comparación con otros temas.

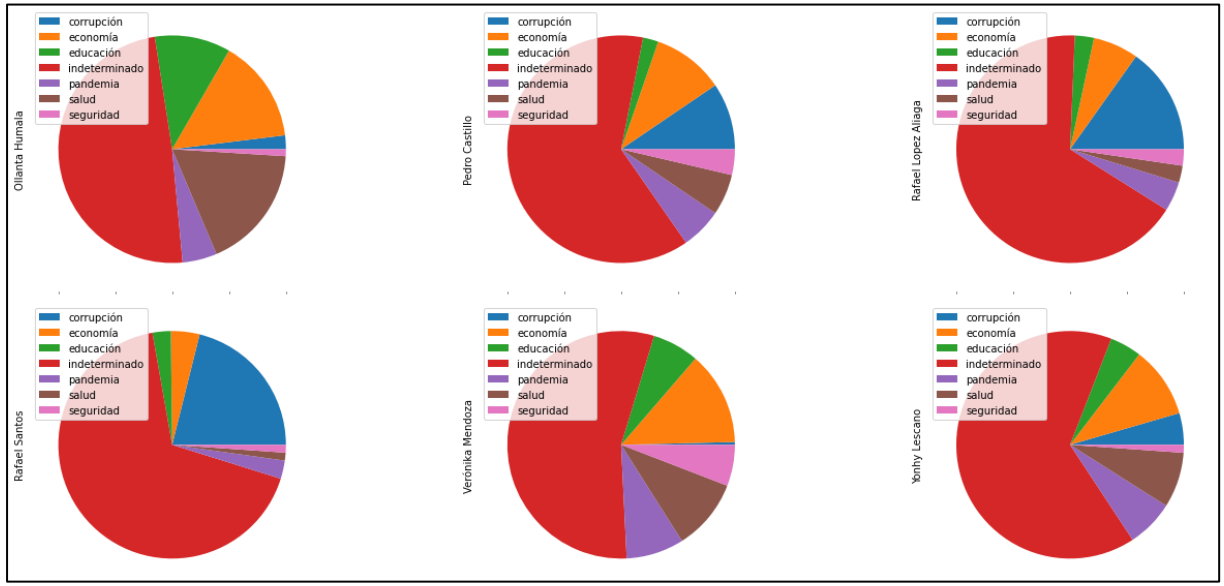
Figura 4.4 Distribución de tópicos por candidato.



Fuente: Elaboración propia

Nota: En la imagen se presenta una comparativa de la distribución de tópicos en los discursos o planes de diversos candidatos políticos. En casi todos los casos, el tópico "indeterminado" (rojo) domina ampliamente la composición, lo que sugiere una falta de definiciones claras o una dificultad del modelo para clasificar gran parte del contenido.
Destacan perfiles específicos como el de George Forsyth, quien muestra una proporción mucho mayor en el tema de corrupción (azul), o César Acuña, cuyo gráfico es de los más equilibrados, con una presencia notable de pandemia (morado) y seguridad (rosa).

Figura 4.5 Distribución de tópicos por candidato.



Fuente: Elaboración propia
Nota: Esta imagen muestra la distribución temática de otros seis candidatos, manteniendo la tendencia donde el tópico "indeterminado" (rojo) ocupa más de la mitad de cada gráfico. Esto confirma que, independientemente del candidato, una gran parte de su discurso no se ajusta estrictamente a las categorías predefinidas.

En la Figura 4.5 se destacan las siguientes observaciones:

- Rafael Santos: Presenta la mayor concentración en el tema de corrupción (azul) de este grupo, con una presencia mínima en otros tópicos identificados.
- Ollanta Humala: Destaca por darle un peso mayor a la salud (marrón) y la economía (naranja) en comparación con sus pares en esta imagen.
- Verónika Mendoza: Muestra un equilibrio notable entre economía (naranja), educación (verde) y seguridad (rosa), siendo de las pocas que resalta este último punto.
- Rafael López Aliaga y Pedro Castillo: Exhiben distribuciones similares con un enfoque moderado en corrupción y economía.

La tabla 4.4 expone los score de kappa para calcular el acuerdo entre pareja de anotadores y todos los anotadores en conjunto. Los score de acuerdo para todos los anotadores en conjunto se calcularon usando el promedio de todos los kappa (Light's kappa). La interpretación es la siguiente:

- **Nivel de acuerdo:** Los valores de Cohen's kappa oscilan entre **0.27 y 0.41**, lo que indica un acuerdo **"aceptable" a "moderado"** (según la escala de Landis y Koch). Esto sugiere que la tarea de clasificar estos tweets es subjetiva o compleja para los humanos.
- **Consistencia:** El par de anotadores **1 y 2** presenta la mayor concordancia (0.41), mientras que el par **2 y 3** es el que más discrepa (0.27).
- **Ajuste de sesgo:** El kappa de Byrt (ajustado) muestra valores más estables (alrededor de **0.37** para el conjunto), lo que compensa posibles desequilibrios en la distribución de las categorías anotadas.

Tabla 4.4: Resumen de score kappa

	Anotadores	**Cohen' kappa**	**Byrt' kappa (ajustado)**
Tweets	1 y 2	0.41	0.36
	2 y 3	0.27	0.36
	1 y 3	0.33	0.40
	Todos	0.34	0.37

Fuente: Elaboración propia
Nota: La tabla presenta un análisis del acuerdo entre anotadores utilizando los coeficientes de Kappa (Cohen y Byrt) para la clasificación de tweets.

De los 3188 (56.04%) tweets en los que los 3 anotadores llegaron a un acuerdo en conjunto, 2899 (50.96%) no ocupan ninguna categoría ("indeterminado") siendo además el tópico más representativo del dataset. Asimismo, 75 (1.32%) tweets de los 3188 pertenecen a "corrupción", 64 (1.12%) a "pandemia", 59 (1.04%) a "economía", 46 (0.81%) a "educación", 33 (0.58%) a "salud" y 22 (0.39%) pertenecen a la categoría "seguridad". Los porcentajes para las categorías definidas son muy bajos debido a que la mayor parte del dataset está poblado por tweets "indeterminados" por los anotadores, estos posteriormente fueron removidos para continuar con el proceso de entrenamiento de los modelos de inteligencia artificial, haciendo que el porcentaje de los tópicos definidos por el estudio queden balanceados.

4.1.3. Clasificación de tópicos políticos

Para la investigación se utilizó diferentes técnicas de clasificación supervisada para abordar el objetivo de esta investigación, desde técnicas tradicionales hasta modelos basados en Deep Learning con configuración de diversos hiper-parámetros.

4.1.3.1. Representación vectorial de los textos

Los algoritmos de Machine Learning y Deep Learning reciben datos de entrada numéricos y ya que se está tratando con texto, es preciso emplear técnicas que permitan transformar los textos en datos numéricos, para esto, se empleó técnicas de vectorización númerica muy usadas en el campo de NLP las cuales son TF-IDF, Word2Vec y FastText. TF-IDF vectoriza cada oración en función del número de ocurrencias de cada palabra en el corpus, además pondera estos resultados usando el factor idf de modo que las palabras con más ocurrencias en el corpus tengan un menor puntaje que las que son menos frecuentes. Los Word Embeddings son algoritmos no supervisados basados en redes neuronales que vectorizan palabras en función a su contexto semántico en el corpus, FastText (Bojanowski et al., 2016), así como Word2Vec (Mikolov et al., 2013) son modelos capaces de este tipo de vectorización, sin embargo, FastText además permite

representar vectorialmente palabras “fuera de vocabulario” (OOV), es decir, palabras que no aparecieron durante la fase de entrenamiento de este modelo.

4.1.3.2. SVM con TF-IDF y Word Embeddings

Support Vector Machine (SVM) es considerado como un baseline (Glavas et al., 2017), (Abercrombie et al., 2019) útil para comparar contra otros modelos más complejos de Deep Learning como las CNN, por lo que se considera este modelo como punto de partida en los experimentos. Debido a que SVM recibe vectores como entrada, las representaciones producto de usar TF-IDF son ingresadas al modelo sin ningun tipo de post-procesamiento, ya que en TF-IDF cada oración o dato de entrada recibe una representación vectorial única. Por otro lado, los Word Embeddings producen una matriz de “vectores palabra” (word-vectors) de dimension $W\ x\ D$ donde W es el número de palabras presentes en una oración o dato de entrada y D es la dimensión del vector semántico que es variable y es definido por el mismo algoritmo. Unir todas las representaciones vectoriales de palabras presentes en cada matriz terminaría por un espacio vectorial muy grande que no sería muy bien aprovechado por SVM, por lo que una mejor estrategia es la de promediar todos los vectores presentes en una matriz para obtener una representación semántica que corresponda a toda la oración. Para esto se calculó un promedio ponderado como en (Glavas et al., 2017):

$$WCBOW(t_1 \dots t_k) = \frac{1}{\sum_{i=1}^{k} w_i} \sum_{i=1}^{k} w_i e(t_i)$$

Donde t_i es el i-nésimo token de la oración o dato de entrada, $\mathrm{e(t_i)}$ es la representación vectorial semántica del token i-nésimo y w_i es un escalar que representa el score de importancia dado por TF-IDF del token i presente en la oración.

De forma alternativa, se calculó el promedio simple usando todos los vectores semánticos presentes en una matriz (Kenter et al., 2015):

$$AVGCBOW(t_1 \dots t_k) = \frac{1}{k} \sum_{i=1}^{k} e(t_i)$$

Donde k es el número de tokens presentes en la oración.

4.1.3.3. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales (CNN) han demostrado ser de mucha utilidad en el ámbito de la clasificación de textos, específicamente, experimentos llevados a cabo en textos políticos (Glavas et al., 2017), (Abercrombie et al., 2019), (Bilbao-Jayo et al., 2018), (Chatsiou, 2020) han demostrado obtener buenos resultados superando a enfoques más tradicionales basados en Machine Learning. Las CNN extraen características ocultas de los datos empleando convoluciones a través de multiplicaciones matriciales, dada una entrada de datos en forma de matriz, los filtros o kernels son los encargados de llevar el proceso de convolución. En el campo de la clasificación de textos se suelen emplear convoluciones de tipo unidimensional, con ello es posible extraer características como los k-grams de una oración donde k representa el tamaño del filtro aplicado en la convolución, de modo que si el tamaño de filtro es 2 (k = 2), entonces se estaría extrayendo los 2-grams o bigramas de la oración.

A diferencia de SVM, las CNN reciben de entrada una matriz, para el caso de esta investigación, esta matriz representa una oración previamente vectorizada mediante una técnica de Word Embeddings, de modo que la matriz de dimensiones W x D es la representación vectorial de un dato de entrada para la CNN de la presente investigación. Sin embargo, W puede variar ya que no todas las oraciones poseen la misma cantidad de tokens y es preciso señalar que la CNN debe recibir como entrada matrices de igual dimensión. Para ello, se llevó a cabo el siguiente proceso: se define N como el número máximo de tokens que una oración pueda contener, a los textos cuya cantidad de tokens sea menor que N se les agregaba posteriormente L vectores de ceros donde L es la cantidad de tokens necesarios para cumplir con la condición, este es un procedimiento conocido como padding. Por otro lado, los textos que sobrepasaban la condición simplemente eran truncados hasta tener N tokens.

Cabe resaltar que para las CNN, solo se hizo uso de Word Embeddings más no de TF-IDF, ya que como se explicó en el punto 3.5.2.2, TF-IDF vectoriza a cada oración

directamente, mientras que con Word Embeddings es posible obtener una representación matricial de toda la oración haciendo esta más apropiada para servir de entrada a arquitecturas basadas en Deep Learning.

4.1.3.4. Redes Neuronales Recurrentes (RNN) y GRU

Las redes neuronales recurrentes (RNN) son arquitecturas que recientemente han sido empleadas también en el campo de clasificación de textos (Koh et al., 2021). Su uso ha llevado a obtener resultados más precisos en cuanto a clasificación de textos políticos incluso superando a los esquemas propuestos usando CNNs. En particular destacan las RNN que emplean el mecanismo conocido como Gated Recurrent Unit o GRU (Cho et al., 2014).

Las RNN actuan como modelos de mezcla (mixing models) tomando una palabra o token dentro de una secuencia de entrada y mezclándola con la palabra que sigue en la oración, el resultado luego se vuelve a mezclar con la siguiente palabra y así sucesivamente hasta llegar al final de la oración. Este procedimiento se realiza de token a token y de izquierda a derecha mimetizando la forma en como los humanos leen una oración. La RNN en estudio recibe como entrada la misma matriz utilizada para el modelo CNN: una matriz de embeddings W x D donde cada fila representa a un token de una secuencia de entrada, de la misma forma, se mantienen las condiciones usadas para el padding y el truncamiento de secuencias. La RNN retorna W vectores de embeddings mezclados de u elementos, donde u es la cantidad de units utilizadas en la arquitectura del modelo. Visto de forma general, el resultado de aplicar este mixing model a la matriz de embeddings es una nueva matriz de embeddings cuyos vectores engloban un mayor contexto de toda la oración, sin embargo, vuelve a estar presente el problema de que si se concatena todos los vectores de embeddings resultantes, el nuevo vector de entrada que se usará para la predicción para las siguientes capas va a resultar siendo uno de dimensiones muy grandes. Para atender este problema, se aplica el mismo procedimiento que con SVM en el punto 3.5.2.2: promediar los vectores de embedding utilizando un promedio simple.

4.1.4. Evaluación

En esta sección se describirá el dataset usado para la etapa de experimentación, así como el preprocesamiento que se empleó, luego se va a explicar la etapa de experimentación a detalle y se va a finalizar con los resultados.

4.1.4.1. Acerca del DATASET

Como se explicó en la Sección 3.5, se emplearon 2 fuentes de datos para la creación del corpus de entrenamiento: Transcripciones de debates presidenciales y tweets de candidatos a la presidencia. Los pasos a seguir para la limpieza de estas dos fuentes de datos fueron los siguientes:

- Remoción de stopwords mediante una lista predefinida y una lista personalizada con las palabras más repetidas del corpus
- Remoción de caracteres especiales como emojis, así como todos aquellos tokens de longitud 1 (de una sola letra) y que no se encuentren previamente en la lista de stopwords
- Paso de todo a minúsculas
- Creación de un diccionario personalizado con abreviaturas comunes del español y también propias del léxico peruano con el cual se reemplazan todas las abreviaturas que se encontraban en el corpus
- Remoción de todas las oraciones que contengan menos de 10 tokens (longitud mínima de tokens >=10)
- Para la fuente de datos Twitter, se reemplazó las menciones a usuarios por el token especial @USER" y las url/links por el token especial "@HTTPURL" (Daniel X. Sun, 2020)

Cabe resaltar que las tildes no se modificaron, así como los hashtags y los números. Así como tampoco se empleó ninguna técnica de preprocesamiento que modifique los tokens tales como lematización o n-grams (los n-gram como tal fueron inferidos cuando

se entrenó a las CNN), no se empleó ningún límite máximo de tokens debido a que este límite luego iba a variar de acuerdo a la matriz de entrada especificada ya sea en las CNN o en las GRU.

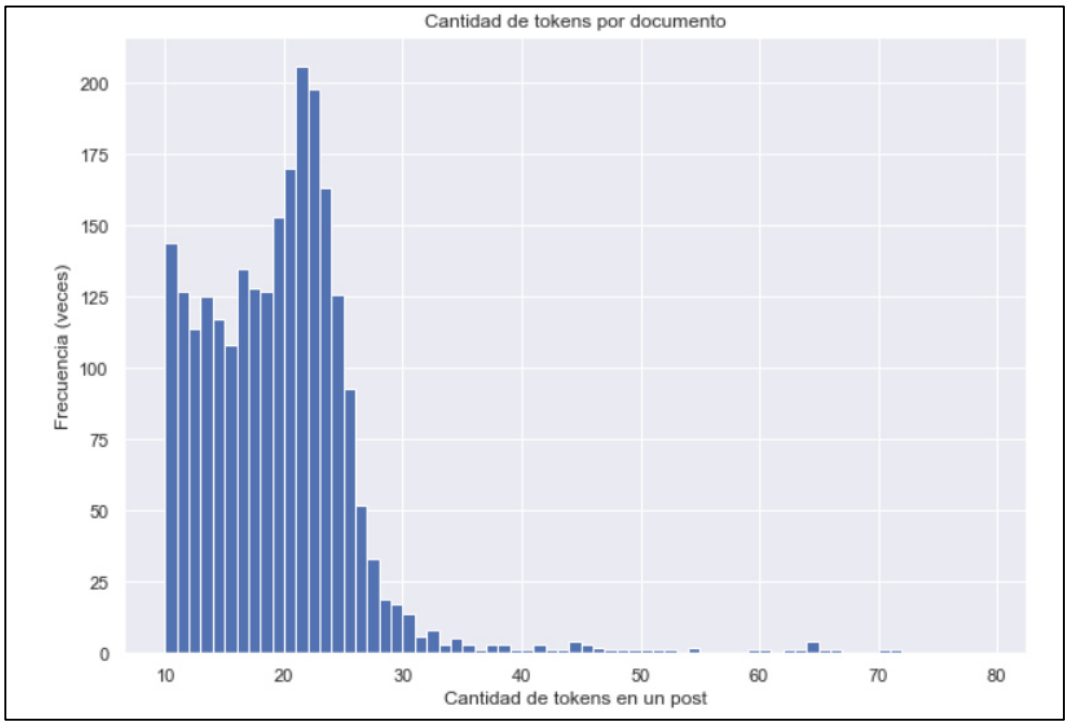
El corpus final de entrenamiento consta de 2439 textos entre tweets y transcripciones de debates políticos (1681 tweets y 758 pseudo-oraciones) y cuenta con las siguiente distribución de categorías políticas: Pandemia (32.6%), Corrupción (20.3%), Seguridad (18.6%), Economía (16.2%) y Educación (12.3%).

El dataset fue dividido siguiendo el ratio 80/20 (1951 textos para training y 488 textos para test), y dentro del 80% que corresponde al training set, un 20% fue separado para servir de validation set. Para SVM, los set de training y validation fueron utilizados para crear modelos usando el enfoque de cross-validation con 5 folds por lo que no se necesitó dividir más el training set, mientras que para los modelos basados en Deep Learning se emplea un solo validation set (20% de training set) para realizar la optimización de los modelos. Finalmente, independiente del método de optimización usado, el test set sirve para probar el rendimiento predictivo final de todos los modelos.

Adicionalmente, el dataset cuenta con 10,328 tokens únicos, 46,590 tokens en general (con repetición) y la cantidad de tokens máxima encontrada en una oración es de 80 tokens, dato que fue utilizado para truncar las representaciones matriciales de las oraciones del corpus según la sección 4.1.3.3.

En la figura 4.6, se observa la distribución de la cantidad de tokens en el corpus final. La mayor cantidad de textos contienen entre 10 a 30 tokens, la cola derecha de la distribución es larga debido a que hay una menor proporción de textos cuya cantidad de tokens supera a la mayoría, gran parte de estos textos corresponde a las transcripciones de debates.

Figura 4.6 Distribución de tokens en corpus final.



Fuente: Elaboración propia
Nota: El histograma expuesto representa la distribución de la longitud de los documentos (en este caso, posts) basada en la cantidad de tokens (palabras o unidades de texto) que contiene cada uno.

Del histograma expuesto en la figura 4.6 se desprende las siguientes conclusiones:

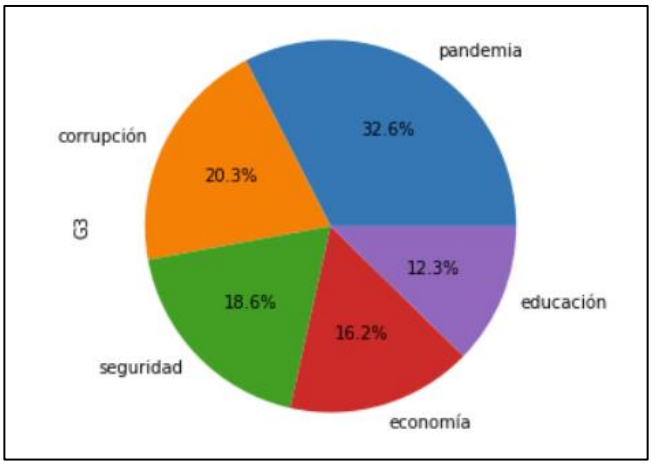
- **Rango de longitud:** La mayoría de los posts tienen una extensión corta, concentrándose principalmente entre los 10 y 30 tokens. Hay un pico muy marcado (moda) alrededor de los 22-23 tokens.
- **Filtro previo:** Se observa que el gráfico comienza exactamente en 10 tokens. Esto confirma la aplicación del filtro de limpieza visto en tus diagramas anteriores (donde se eliminaban registros con menos de 10 tokens).
- **Sesgo a la derecha:** El gráfico presenta una "cola" hacia la derecha, lo que indica que hay muy pocos posts largos (más de 40 tokens), siendo casi inexistentes aquellos que superan los 70 tokens.

- **Volumen de datos:** La frecuencia máxima supera los 200 casos para la longitud de 22 tokens, lo que sugiere un dataset robusto para entrenamiento en ese rango de extensión.

Esta distribución es ideal para modelos de **Deep Learning** (como la CNN o GRU de tus otros diagramas), ya que permite definir un tamaño de secuencia (N) eficiente sin desperdiciar demasiado espacio en *padding* (relleno de ceros).

En la figura 4.7, se observa la distribución de las categorías políticas (en %) del corpus/dataset luego de haber aplicado la limpieza de texto correspondiente.

Figura 4.7 Distribución de categorías en corpus final en porcentaje.



Fuente: Elaboración propia
Nota: En el gráfico se presenta la distribución de tópicos mucho más equilibrada y específica que las anteriores, ya que ha desaparecido la categoría de "indeterminado".

Respecto a lo reflejado en la figura 4.7, se desprende la siguiente interpretación:

- Tópico predominante: La pandemia es el tema principal, ocupando casi un tercio del total con un 32.6%.
- Temas secundarios de alto impacto: La corrupción (20.3%) y la seguridad (18.6%) completan el podio, sumando casi el 40% del discurso analizado.
- Temas económicos y sociales: La economía representa un 16.2%, mientras que la educación es el tema con menor presencia en este gráfico específico con un 12.3%.

4.1.5. Configuración de los experimentos

4.1.5.1. Word embeddings

Se empleó modelos de word embeddings preentrenados para la representación vectorial de cada uno de los tokens del corpus, la razón por la cual no se entrenó el corpus preliminar de embedding fue por la cantidad de datos disponibles, se necesitan grandes cantidades de texto para lograr tener un modelo de embedding robusto. La ventaja de usar modelos preentrenados es que la mayoría de estos ya fue entrenado con grandes cantidades de datos y usualmente ya vienen disponibles para un idioma específico. En el proceso de investigación, se usó un modelo de Word2Vec preentrenado (SBWCE, 2021) con aproximadamente 1.5 billones de palabras en español extraídas de diversos textos de distintos dominios y cuya dimensión final embebida es de 300. También se usó un modelo de FastText preentrenado (FastText, 2021) con aproximadamente 3 billones de palabras extraidas del corpus conocido como Spanish Unannotated Corpora (SUC, 2021) y cuya dimensión final embebida es de 100. Aunque este último embedding estaba disponible en más dimensiones, se elige 100 por razones relacionadas a recursos computacionales.

Si bien los embeddings descritos contienen un vocabulario muy grande, aún así se encontraron palabras en el corpus que no se encontraban en el vocabulario de los embeddings o también conocidas como “OOV (Out-of-Vocabulary) words”. Las OOV más resaltantes eran los hashtags, algunos apellidos de algunos candidatos a la presidencia y algunas palabras relacionadas al dominio político peruano, fuera de ello y dado que estos embeddings preentrenados usan fuentes de dominios muy diversos, es probable que algunas palabras sean embebidas bajo un contexto distinto al campo político. Para esto se llevó a cabo una estrategia de finetuning que involucra actualizar los pesos de estos modelos preentrenados usando fuentes de texto de dominio político peruano:

- Se extrajo 19 planes de gobiernos en formato pdf correspondientes a los partidos políticos postulantes a la presidencia de Perú 2021. En específico, se llegaron a extraer 17,673 pseudo-oraciones luego de un proceso de limpieza de texto. La tabla 4.5 se

muestra en detalle la relación de partidos políticos considerados y la cantidad de pseudo-oraciones obtenidas por cada plan de gobierno luego de haber aplicado la limpieza.

- Se extrajo alrededor de 600,000 tweets relacionados a las elecciones a la presidencia de Perú 2021, los tweets fueron extraídos usando los hashtags tendencia durante la época de elecciones a modo de keywords y corresponden a cualquier usuario de twitter que haya usado tales hashtags. Luego de un proceso de limpieza, el dataset final contó con 347,555 tweets.

- El proceso de limpieza fue el mismo (mismas condiciones y parámetros) que se utilizó para el preprocesamiento del corpus de entrenamiento descrito en el punto 4.1.4

Tabla 4.5: Relación de partidos considerados en la investigación

Partido Político/ Plan de gobierno	Cantidad de pseudo-oraciones	Cantidad de tokens
Accion popular	332	66,105
Alianza para el progreso	3,032	522,547
Democracia directa	527	98,840
Frente amplio	994	147,924
Fuerza popular	1,080	189,439
Juntos por el peru	1,146	169,342
Partido aprista peruano	653	80,182
Partido morado	1,682	296,199
Partido nacionalista	792	118,992
Partido popular cristiano	210	40,660
Peru libre	746	146,369
Peru patria segura	2,078	369,100
Renovacion popular	296	43,148
Somos peru	1,987	333,861
Union por el peru	338	67,994
Victoria nacional	1,780	279,653
Total	17,673	2,970,355

Fuente: Elaboración propia

Nota: Esta tabla presenta las métricas de rendimiento de cuatro modelos de clasificación (SVM, CNN, CNN Simple y GRU) bajo dos configuraciones distintas: Bow/Tf-idf (clásica) y Embedding (semántica).

4.1.6. SVM

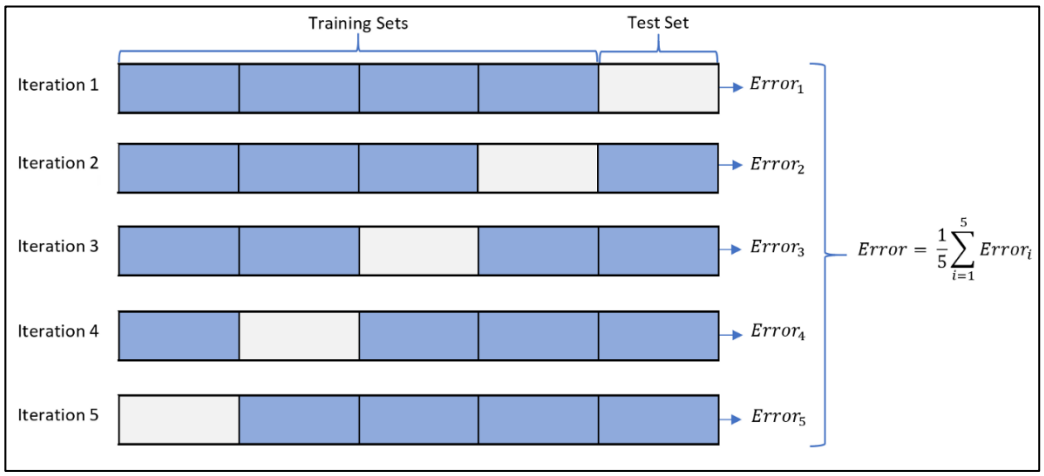
De acuerdo con lo mencionado en la Seccion 4.1.4, se empleó todas las técnicas de vectorización descritas entre TF-IDF y Word embeddings. SVM representa el modelo baseline que sirvió para compararlo contra los enfoques basados en Deep Learning, además de ello, este fue el único modelo que utilizó la vectorización por TF-IDF o Bag Of Words debido a que su formato de salida se adapta muy bien al modelo (cada oración posee una representación vectorial única). Por otro lado, en cuanto a los embeddings, los dos enfoques (promedio ponderado y promedio simple) mencionados en la Seccion 4.1.4 fueron utilizados para fines de comparación.

Para el entrenamiento, se usó la técnica de cross-validation con 5 folds (ver figura 4.9), esta técnica se basa en dividir aleatoriamente el training set en un subset de training y validation por k cantidad de folds. Cross-validation proporciona una forma robusta de medir el rendimiento predictivo esperado de un modelo. Pese a ser una técnica computacionalmente compleja ya que el proceso de entrenamiento se tiene que repetir k veces, esto no representó un problema debido a la cantidad limitada de datos del corpus (2.4k textos).

Asimismo, se empleó un SVM con kernel radial (RBF) debido a la cantidad de datos del corpus (Glavas et al., 2017). Además se usó dos versiones de la técnica de BoW (Bag of Words): TF-IDF y matriz de frecuencias. La diferencia entre estos dos enfoques es que TF-IDF multiplica la matriz de frecuencias obtenida del corpus con el factor IDF (Inverse Document Frequency) obtenido haciendo que la matriz de frecuencias ya no solamente esté conformada por ceros y unos. En cuanto a los hiperparámetros: C=1.0 y gamma=0.77.

En la Figura 4.8, se expone el esquema de representación simbólica del proceso de cross-validation con 5 folds. El proceso de entrenamiento se debe repetir 5 veces y cada una de las 5 iteraciones presenta un test (validation) distinto. En cada iteración se calcula un error de predicción y la métrica final de todo el proceso es un promedio simple de todos los errores.

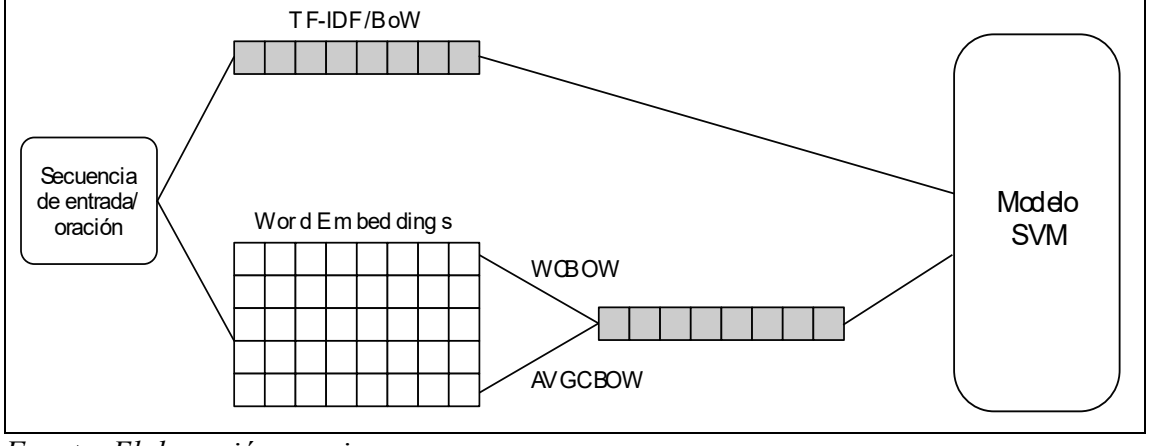
Figura 4.8 Esquema cross-validation.



Fuente: Elaboración propia
Nota: Esta tabla es el resumen comparativo final que evalúa el rendimiento de diversos algoritmos de Machine Learning y Deep Learning aplicados a la clasificación de texto.

En la figura 4.9, se expone el flujo del modelo SVM junto con los dos tipos de representación vectorial utilizados (BoW y Word Embeddings). Se observa como la vectorización mediante word embeddings da como resultado una matriz que luego es procesada mediante un promedio (WCBOW o AVGCBOW) y cuyo resultado es la entrada del modelo SVM. Para fines explicativos, la matriz resultante corresponde a un embedding de 8 dimensiones y a una secuencia de entrada de 5 tokens (sin padding).

Figura 4.9 Flujo del modelo SVM



Fuente: Elaboración propia

Nota: Se presenta el flujo del modelo SVM usando dos representaciones: BoW y Word Embeddings. En embeddings, la matriz generada se promedia (WCBOW/AVGCBOW) y el resultado se usa como entrada del SVM.

4.1.6.1. CNN

La arquitectura de redes neuronales convolucionales (CNN) utilizada en este estudio está inspirada en la propuesta de (Bilbao-Jayo et al., 2018). Específicamente esta arquitectura fue diseñada para clasificar textos políticos provenientes de un corpus llamada Regional Manifesto Project y posteriormente fue probado en un caso de aplicación usando un dataset de tweets sobre la campaña política española. Los autores lograron resultados comparables al estado del arte en clasificación de textos políticos, razón por la cual se elige para experimentar con el corpus. Paralelamente, se experimentó con una arquitectura CNN más simple con solo una capa convolucional.

Para el entrenamiento se utilizó la técnica de Early Stopping (Prechelt, 1998) que consiste en monitorear el score resultante de la función loss de modo que el entrenamiento se detenga cuando no haya una mejora considerable en tal score luego de un determinado número de épocas de tolerancia. Esta técnica reduce los tiempos de entrenamiento así como ayuda a reducir el overfitting en el modelo. La tolerancia aplicada en este estudio fue de 10 épocas. La función loss categorical cross-entropy fue usada en el entrenamiento ya que soporta clasificación multiclase, mientras que el algoritmo de gradiente descendiente fue optimizado usando Adam (Kingma y Ba, 2014).

4.1.6.1.1. CNN

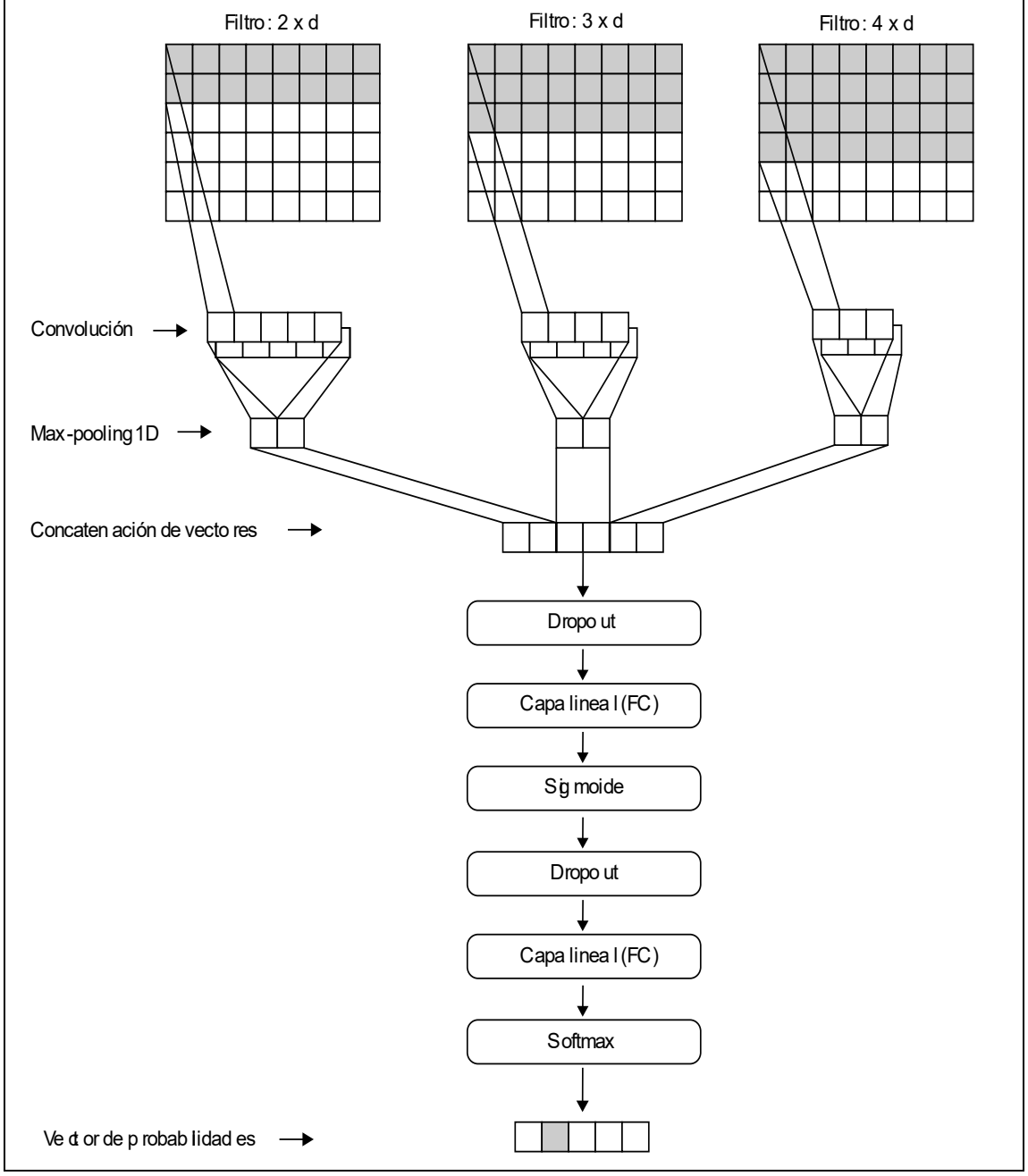
Esta arquitectura consta de 3 tipos de filtros de distintos tamaños expresados en matrices de dimensiones: *2 x d, 3 x d y 4 x d* donde *d* representa el número de columnas de la matriz o la dimensión del embedding. Cada uno de estos filtros es usado dentro de una capa convolucional con el objetivo de extraer los n-grams de la secuencia/oración de entrada, por ejemplo, el filtro de tamaño 2 *x d* extrae los 2-grams o bigramas de la oración. Cabe resaltar que para las CNNs de esta sección solo se experimentó usando embeddings

y siguiendo los procedimientos mencionados en la sección 3.5.2.3. En la arquitectura original, los autores emplean 100 filtros por cada tamaño, es decir emplean 3 capas convolucionales, donde cada una de estas capas consta de 100 filtros. Luego del procedimiento de convolución, los autores emplearon un max-pooling unidimensional de modo que de cada convolución salga solo un número, el cuál, representa a la feature o característica más representativa, este resultado varía de acuerdo al número de filtros utilizados por cada capa convolucional, por ejemplo, de una sola capa convolucional de 100 filtros a la cual se le aplica max-pooling, el resultado final sería un vector de 100 elementos, esta misma lógica se aplica a cada capa convolucional en la arquitectura de modo que al final se obtenga un set de vectores de características que pasan a ser concatenados para formar un solo vector de características más grande que luego sirve de entrada a una capa lineal o Fully Connected (FC) con una función de activación softmax.

En la investigación, se optimiza esta arquitectura mediante el uso de un finetunning de hiperparamentros, como dato relevante, también se experimenta con el número de capas lineales (FC) antes de aplicar softmax. Como resultado, la arquitectura consistió en 3 capas convolucionales con función de activación reLU, cada una de estas capas cuenta con 150 filtros, los tamaños de filtro usados fueron 2 x d, 3 x d y 4 x d, los cuales actúan como extractores de n-grams de la oración/secuencia de entrada. Luego del proceso de convolución, se aplicó un max-pooling unidimensional a los vectores resultantes, dando como resultado un vector de tamaño n por cada capa convolucional, donde n representa al número de filtros correspondiente a cada capa. Seguidamente, los vectores resultantes fueron concatenados dando resultado a un solo vector de características más grande, este vector serviría de dato de entrada para una capa lineal (FC) de 512 neuronas con función de activación sigmoide. Finalmente, el resultado pasaría por una siguiente capa lineal de 5 neuronas (la misma cantidad de categorías a predecir) con función de activación softmax. De esta forma, la salida de la arquitectura CNN es un vector de probabilidades conformado por 5 elementos, el elemento con la probabilidad más cercana a 1 en este vector de salida es la categoría predicha por el modelo. Cabe resaltar que se empleó un dropout de 0.5 antes de cada capa lineal, así como un learning rate de 0.001 y no se aplicó batch normalization en la arquitectura.

En la figura 4.10 se expone la arquitectura propuesta basada en el modelo de (Bilbao-Jayo et al., 2018), en esta se muestra un ejemplo simbólico usando solo 2 filtros por cada capa convolucional y una secuencia de entrada de 6 tokens representada mediante embeddings en una matriz de 6x8 donde 8 es la dimensión del embedding.

Figura 4.10 Arquitectura propuesta por Bilbao & Jayo.



Fuente: Adaptado de Bilbao-Jayo et al., (2018)

Nota: Se ilustra un ejemplo con 2 filtros por capa convolucional y una secuencia de 6 tokens representados como embeddings en una matriz 6×8.

4.1.6.1.2. CNN simple

Se implementó una arquitectura reducida con respecto a la arquitectura descrita en el punto inmediatamente anterior. El objetivo era procurar identificar si la complejidad de la arquitectura afectaba al entrenamiento del modelo con el corpus final, ya que este cuenta con una cantidad limitada de texto y usualmente las arquitecturas de Deep Learning requieren de grandes cantidades de datos para producir resultados más precisos. La reducción aplicada a esta arquitectura CNN fue en la cantidad de capas convolucionales, en lugar de usar 3 capas paralelas que extraian los n-grams de las secuencias de entrada, se prefirió usar solo una capa convolucional con el mismo objetivo, en este caso, el tamaño del filtro aplicado fue elegido mediante un proceso de finetunning.

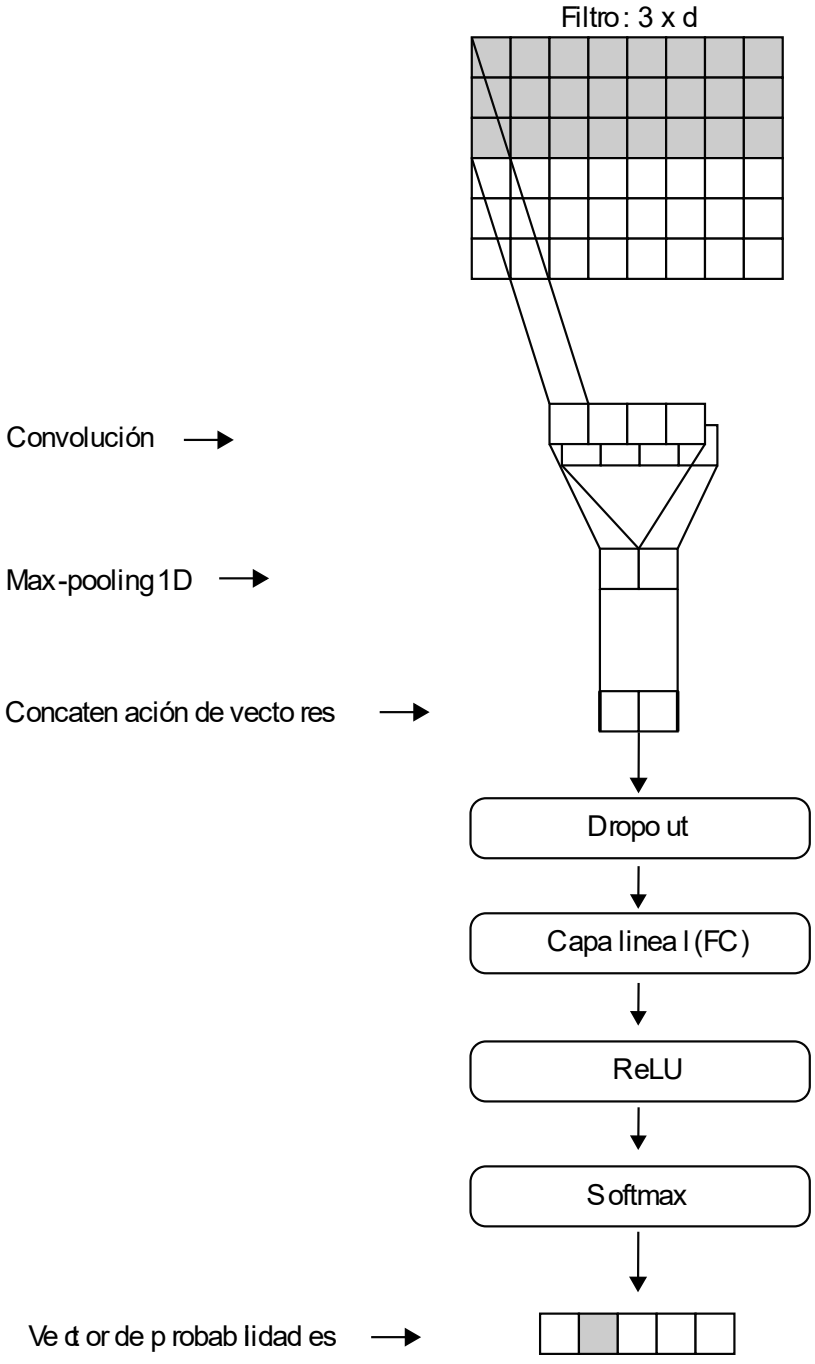
Esta arquitectura consiste en una capa convolucional con 100 filtros de tamaño de filtro 3 x d (tri-grams). La capa convolucional utiliza una función de activación reLU. Luego del proceso de convolución se aplica una capa de max-pooling unidimensional dando como resultado a un vector de características de 100 dimensiones. Luego este vector resultante sirve de entrada a una capa lineal (FC) de 5 neuronas, se utilizaron dos funciones de activación, primero reLU cuyo resultado fue evaluado seguidamente con softmax para finalmente dar un vector resultante de probabilidades. A diferencia de la arquitectura descrita en la figura 4.10, no se hizo uso de capas lineales intermedias (hidden layers). Se utilizó un dropout de 0.25 antes de la capa lineal y el learning rate fue de 0.001. Cabe resaltar que de forma similar a la arquitectura descrita en la figura 4.10 tampoco se empleó el procedimiento de batch normalization a los vectores resultantes de la multiplicación de pesos en la capa lineal.

De igual manera, en la figura 4.11, se muestra el diagrama de esta arquitectura utilizando un número de filtros simbólico a modo de ejemplo.

En la figura 4.11, se muestra la arquitectura propuesta simplificada basada en el modelo de (Bilbao-Jayo et al., 2018). La capa convolucional cuenta con un tamaño de

filtro de 3 x d, capaz de extraer los tri-grams de la oración de entrada. Tal cual como en la figura 4.11, a modo de ejemplo en esta, la capa convolucional cuenta con solo 2 filtros y una secuencia de entrada de 6 tokens vectorizada usando embeddings de dimensión 8.

Figura 4.11Arquitectura simplificada tomando como referencia la arquitectura original propuesta por Bilbao & Jayo.



Fuente: Elaboración propia

Nota: La figura muestra un ejemplo con una capa convolucional de 2 filtros. La entrada es una secuencia de 6 tokens vectorizados en embeddings de dimensión 8.

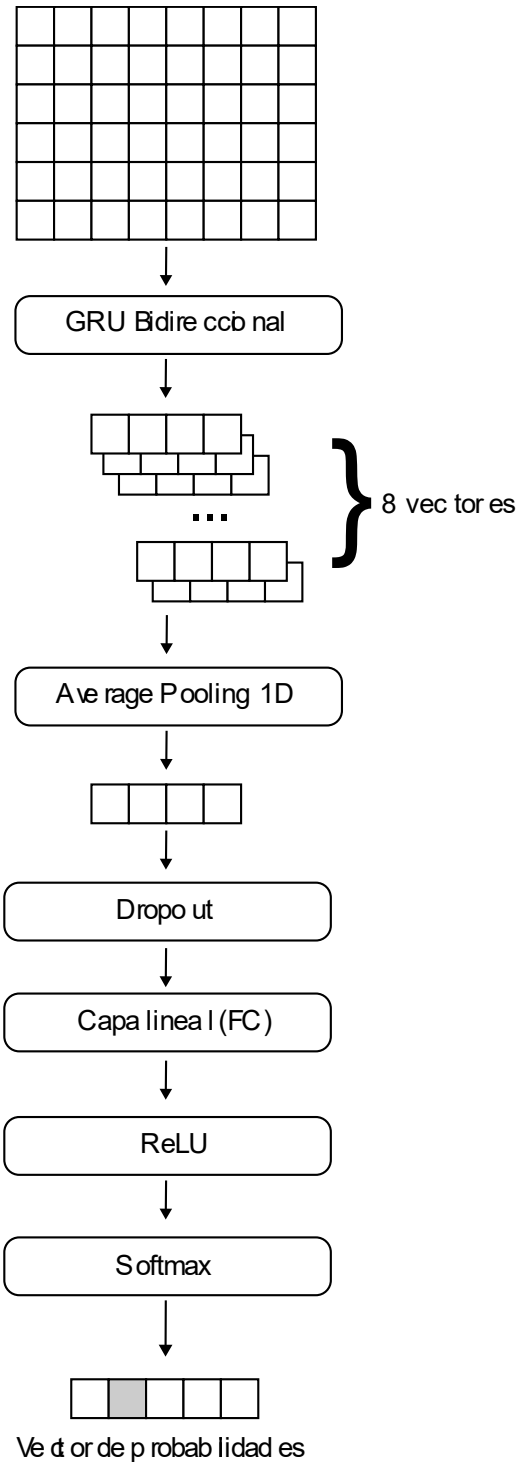
4.1.6.2. RNN

Se implementó una arquitectura basada en redes neuronales recurrentes (RNN) inspirada en el trabajo de (Koh et al., 2021) que hace uso de una RNN con el mecanismo GRU y una sola capa lineal con softmax para la predicción multiclase. En el caso de la investigación emprendida, la arquitectura consiste en una RNN de 32 units con GRU bidireccional, es decir, la mezcla entre embeddings no solo se realiza de izquierda a derecha sino del modo opuesto también (derecha a izquierda), se empleó este tipo de mezcla esperando obtener embeddings que engloben un mayor contexto incluso mejor que mezclando solo en una sola dirección. El resultado es promediado usando un promedio simple entre todos los nuevos embeddings para finalmente obtener un solo vector que englobe el contexto de toda la secuencia de entrada. Este vector luego es ingresado a una capa lineal o Fully Connected (FC) de 5 neuronas con un dropout de 0.25 y dos funciones de activación: primero se aplicó reLU al resultado de la multiplicación entre pesos y luego se aplicó softmax. El resultado final de la arquitectura es una vector de probabilidades de 5 elementos.

Se utilizó un learning rate de 0.01 y al igual que los modelos anteriormente explicados, no se utilizó la técnica de batch normalization.

En la figura 4.12, se expone la arquitectura basada en RNN usando una capa bidireccional GRU y una capa lineal softmax para la predicción. La figura muestra como ejemplo una matriz de embeddings de 6 filas (6 tokens) y 8 columnas (embedding de dimensión 8), así como una capa GRU bidireccional de 4 units, razón por la cual los vectores resultantes de aplicar la capa GRU son de tamaño 4.

Figura 4.12 Arquitectura basada en RNN.



Fuente: Elaborción propia

Nota: En la figura se muestra una RNN con GRU bidireccional y una capa lineal softmax para predicción. Como ejemplo, la entrada son 6 tokens con embeddings de 8 dimensiones y la GRU produce vectores de tamaño 4.

4.1.7. Optimización de hiperparámetros

Las configuraciones de los modelos descritos en las secciones anteriores fueron resultado de una optimización de hiperparámetros previa, se utiliza el optimizador Hyperband (Li y Jamieson, 2018) junto con Early Stopping para los modelos basados en Deep Learning (CNN y RNN) de modo que los tiempos de entrenamiento de cada modelo a probar no sea demasiado. Mientras que para SVM se utiliza el algoritmo GridSearch.

En lo que respecta a SVM el espacio búsqueda comprendió a los hiperparámetros C: un espacio logarítmico de base 10 entre -1 a 2 que comprendía una secuencia de10 muestras entre 10^{-1} hasta10^{2}, gamma: un espacio logarítmico de base 10 entre -1 a 1 que comprendía una secuencia de 10 muestras entre 10^{-1} hasta 10^{1} y también se experimentó usando tf-idf o solo la matriz de ocurrencias de BoW.

Para la CNN inspirada en (Bilbao-Jayo et al., 2018), el espacio de búsqueda compredió al número de capas convoluciones: entre 1 a 3, el número de filtros, la cantidad de capas lineales a utilizar, sus funciones de activación (entre reLu y sigmoide) y la cantidad de neuronas en cada capa, si utilizar reLU antes de softmax, si utilizar batch normalization antes de cada función de activación y el ratio de dropout que variaba entre 0.0 a 0.8 donde 0.0 significaba no usar dropout.

Para la CNN versión simple, el espacio de búsqueda fue similar al empleado para la versión de CNN utilizada, la única variación fue en cuanto a las capas convolucionales, ya que esta arquitectura solo usa 1 capa convolucional, el espacio de búsqueda se reducía a variar el tamaño del filtro a utilizar, este podía variar entre las tres versiones mencionadas anteriormente: 2 x d, 3 x d y 4 x d.

Cabe resaltar que por cada conjunto de hiperparámetros también se realizó pruebas de distintos límites máximos de tokens para una secuencia de entrada (Sección 3.5.4.3): 30, 40, 50, 60, 65, 70 y 80 tokens como máximo, estas cantidades fueron determinadas después de un análisis de la longitud de oraciones del corpus de la investigación, la oración más larga contiene 80 tokens. De acuerdo con el límite máximo elegido, el tiempo de

entrenamiento y optimización del modelo variaba ya que el tamaño de filas de las matrices se ve afectado, por ejemplo, con un límite de 30 tokens, se tendría una matriz de 30 filas y un tiempo de entrenamiento y optimización menor comparado a una configuración con un límite de 80 tokens.

4.1.8. Análisis de resultados

En relación con los objetivos de esta investigación, se busca medir el rendimiento de un modelo predictivo capaz de clasificar textos políticos en 5 categorías distintas, en particular, se prestó especial atención en medir este nivel de predicción (1) en un modelo que entrene con un corpus variado que comprende una mezcla entre tweets y textos de debates políticos, (2) en un modelo entrenado con solo tweets para clasificar tweets y (3) en un modelo entrenado con esos mismos tweets pero esta vez para clasificar las pseudo-oraciones de los debates políticos. Los resultados obtenidos producto de los experimentos y de la optimización de hiperparámetros se encuentran plasmados en las tablas 4.6, 4.7 y 4.8. Se utilizó dos métricas de evaluación: el F1-score micro y el F1-score macro con el objetivo de comparar los resultados con el estado del arte en arquitecturas de clasificación de textos políticos, cabe resaltar que si bien la fuente de datos difiere con lo usado en el estado del arte, comparar los resultados aún nos sigue siendo útil a modo de baseline para conocer si los modelos propuestos van en la dirección correcta.

En la tabla 4.6 se exponen los resultados de los modelos en términos de F1 Score (Macro y Micro) utilizando como fuente de entrenamiento al corpus descrito en la Sección 3.5.3.1 que comprende una mezcla entre tweets y pseudo-oraciones extraídas de debates electorales. Los número en negrita reflejan el mejor resultado por cada tipo de F1-Score.

Tabla 4.6: Resultados de modelos F1Score (Macro y Micro)

Modelo	**Representación de texto**	**Fuente de entrenamiento**	**Evaluación (F1 Score)**	
			Macro	Micro
SVM	BoW	Twitter+debate	0.6538	0.6701
	Word emb (w2v)		0.2757	0.3955
	Word emb (fastText)		0.6776	0.6896
	Word emb (fastText finetuning)		0.7236	0.7295
CNN (1)	Word emb (fastText finetuning)		0.7356	0.7299
CNN (2)	Word emb (fastText finetuning)		0.7377	**0.7351**
RNN	Word emb (fastText finetuning)		**0.7459**	0.7334

Fuente: (Bilbao-Jayo et al., 2018) Multicapa convolucional y simple
Nota: Los resultados muestran que los modelos de deep learning (CNN y RNN) superan al SVM en la clasificación de texto. El uso de embeddings fastText con finetuning mejora notablemente el rendimiento frente a otras representaciones. El modelo RNN obtiene el mejor desempeño, alcanzando el F1 macro más alto (0.7459).

Tabla 4.7: Resultados de los modelos en términos de F1 Score – Corpus Tweets

Modelo	**Representación de texto**	**Fuente de entrenamiento**	**Evaluación (F1 Score)**	
			Macro	Micro
SVM	BoW	Twitter (Test en Twitter)	0.6365	0.6855
	Word emb (w2v)		0.2772	0.4154
	Word emb (fastText finetuning)		0.6699	**0.727**
CNN (Bilbao-Jayo et al., 2018)	Word emb (fastText finetuning)		0.7537	0.6964
CNN (simple)	Word emb (fastText finetuning)		**0.7596**	0.7065
RNN	Word emb (fastText finetuning)		0.7388	0.6914

Fuente: Elaboración propia
Nota: Los resultados muestran que los modelos neuronales (CNN y RNN) obtienen mejor desempeño que SVM en el conjunto de prueba de Twitter. La representación fastText con finetuning mejora claramente el rendimiento frente a BoW y Word2Vec. El mejor resultado lo obtiene la CNN simple, con el mayor F1 macro (0.7596).

En la tabla 4.8 se exponen los resultados de los modelos en términos de F1 core (Macro y Micro) utilizando como fuente de entrenamiento un corpus que comprende solo tweets. En este punto se empleó los mismos modelos entrenados de la tabla 4.6 los cuales fueron evaluados con el corpus de debates electorales (en lugar de tweets) para conocer sus rendimientos al utilizar un dataset con textos de contexto y escritura distinta.

Tabla 4.8: Resultado de los modelos F1 score (Solo Tweets)

Modelo	**Representación de texto**	**Fuente de entrenamiento**	**Evaluación (F1 Score)**	
			Macro	Micro
SVM	BoW	Twitter (Test en Debate)	0.4364	0.4683
	Word emb (w2v)		0.1381	0.3455
	Word emb (fastText finetuning)		0.4852	0.5277
CNN (Bilbao-Jayo et al., 2018)	Word emb (fastText finetuning)		**0.5923**	**0.5311**
CNN (simple)	Word emb (fastText finetuning)		0.5527	0.5029
RNN	Word emb (fastText finetuning)		0.5580	0.5022

Fuente: Elaboración propia
Nota: Los resultados muestran una disminución del rendimiento cuando el modelo entrenado con Twitter se evalúa en datos de debate, lo que indica diferencias entre dominios. Los modelos neuronales (especialmente la CNN de Bilbao-Jayo) obtienen los mejores resultados, superando a SVM. Además, fastText con finetuning sigue siendo la representación que logra mejor desempeño.

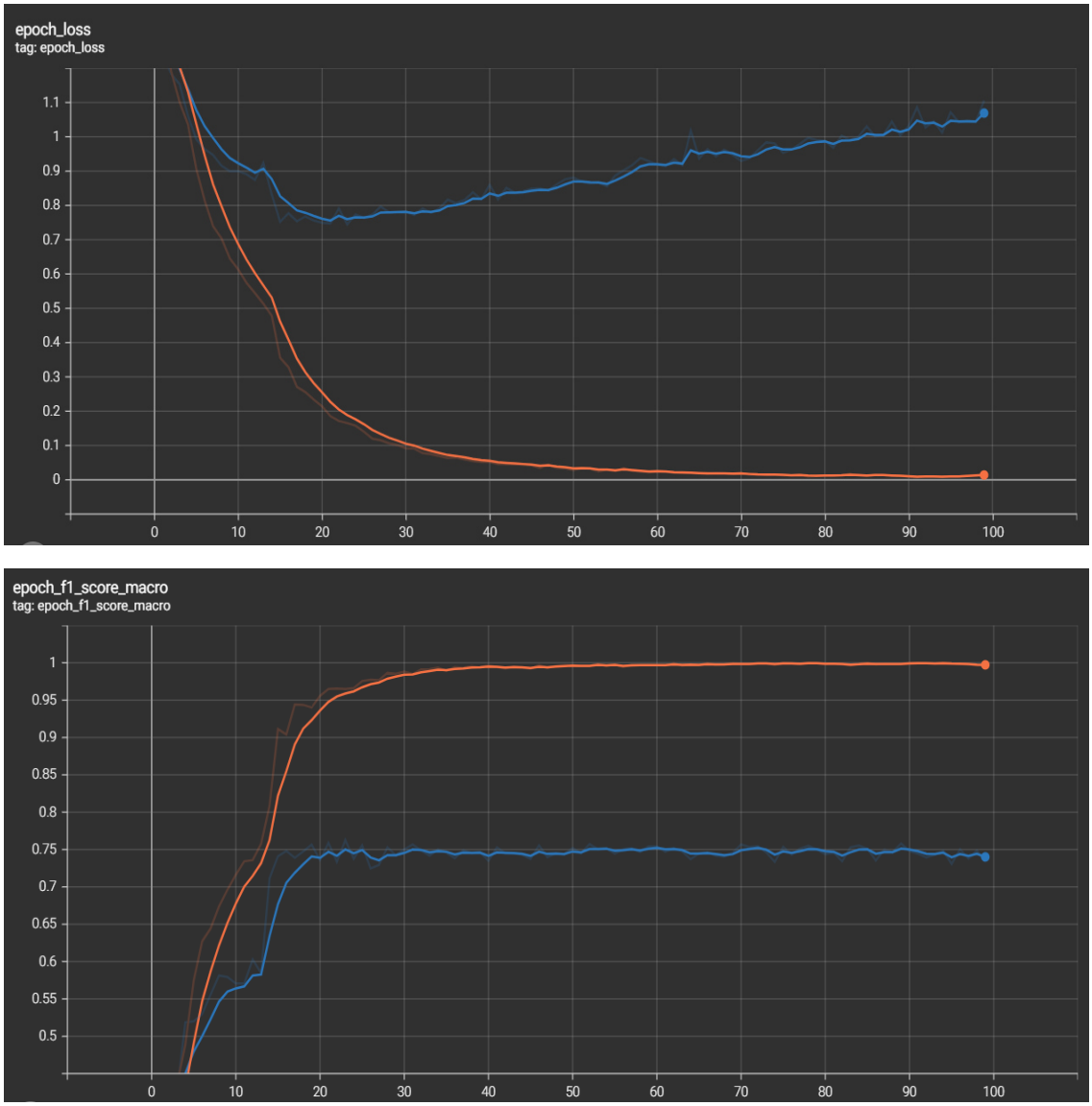
En vista que las categorías de los modelos que clasifican están desbalanceadas se tomó como métrica principal al F1-score macro, el cuál se adapta mejor a este tipo de problemas, sin embargo, tambíen se calculó el F1-score micro con fines de comparación. De esta forma, se puede decir, que según la tabla 4.8, el mejor modelo para clasificar textos políticos que comprenden fuentes variadas entre tweets y pseudo-oraciones de debates electorales es la arquitectura basada en Redes Neuronales Recurrentes, aunque en sí se observa que la diferencia es mínima e incluso el modelo baseline se acerca al score obtenido por el modelo RNN.

Resalta en el proceso de investigación las dos mejores arquitecturas para este objetivo: CNN y RNN obteniendo puntajes muy similares en cuestión de F1-score macro. De los experimentos realizados con las combinaciones de CNNs se observa como a medida que el límite máximo de tokens dismunuía, el modelo se hacia más inestable, es decir, las curvas de validation presentabas fluctuaciones a medida que avanzaban las épocas. Este fenómeno se observó en menor medida utilizando el límite máximo de 60 tokens que vendría a estar dentro del punto medio dentro de la distribución de tokens por texto del corpus de la investigación. Sin embargo, pasado el límite de 60 también se empezaban a notar de nuevo este tipo de fluctuaciones aunque de menor notoriedad en comparación de las anteriores obtenidas con límites menores a 30 tokens.

En la figura 4.13, se muestra la curva de entrenamiento y loss para la CNN simple y la figura 4.14 muestra las mismas curvas de la misma arquitectura pero bajo un límite máximo de 30 tokens. Además de la diferencia de fluctuaciones, también se observó que la CNN bajo la configuración de límite 60 converge más rápido que las demás configuraciones de CNN (alrededor de 50 épocas para llegar a un training loss cercano a 0.1).

En la Figura 4.13 se observa las curvas de entrenamiento y loss del modelo CNN simple utilizando un límite máximo de 60 tokens y entrenado a 100 épocas. La curva azul representa la curva de validation y la curva naranja representa la curva de training.

Figura 4.13 Curva de entrenaniento y loss de modelo CNN.

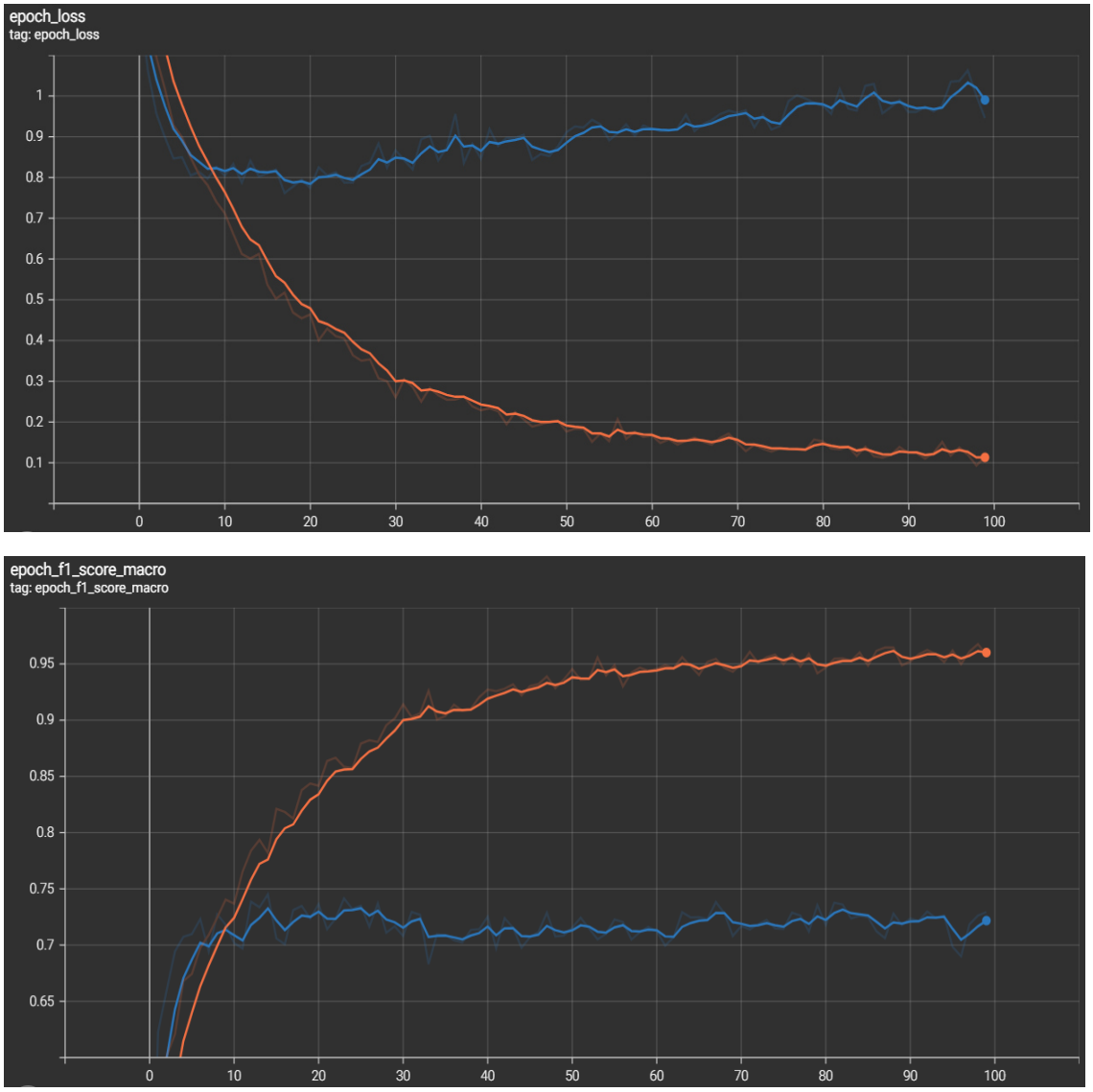


Fuente: Elaboración propia
Nota: La figura muestra la curva de entrenamiento y la función de pérdida (loss) del modelo CNN durante el proceso de aprendizaje. Se observa cómo el loss disminuye progresivamente, indicando que el modelo mejora su ajuste a los datos. Esto sugiere que el entrenamiento converge y el modelo aprende patrones del conjunto de entrenamiento.

En la figura 4.14 se observa las curvas de entrenamiento y loss del modelo CNN simple utilizando un límite máximo de 30 tokens y entrenado a 100 épocas. La curva azul representa la curva de validation y la curva naranja representa la curva de training. Las

fluctuaciones de la curva son más notorias lo cuál la vuelve una arquitectura más inestable bajo este tipo de configuración.

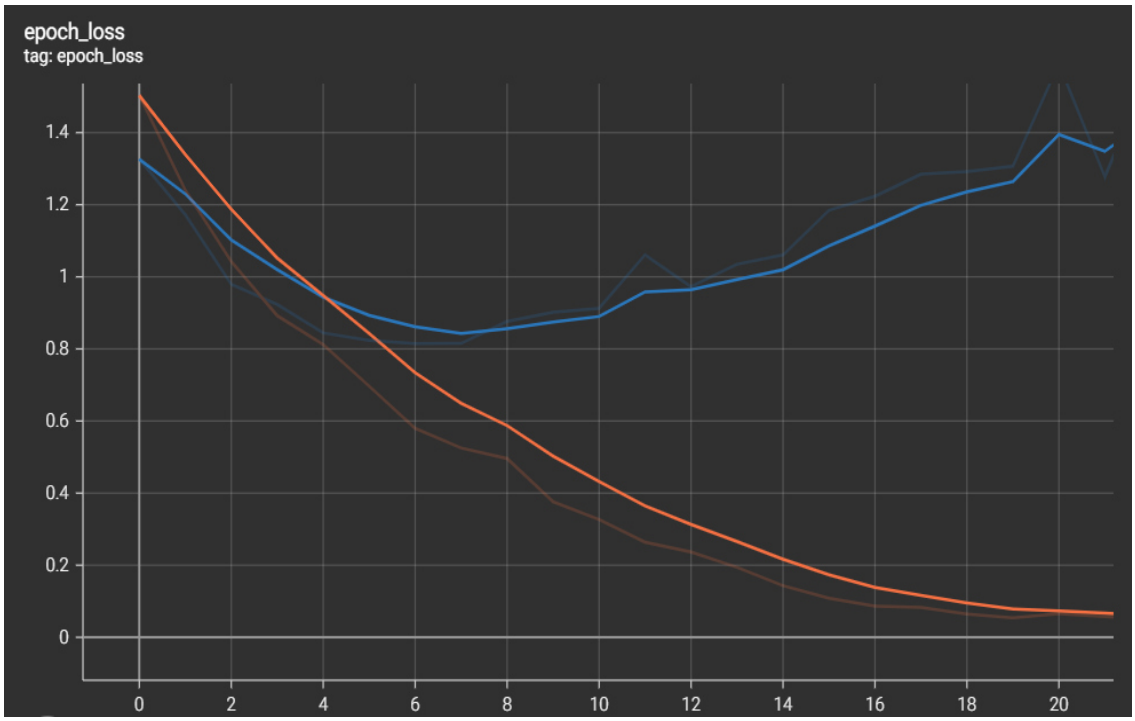
Figura 4.14 Curva de entrenamiento y loss del modelo CNN.

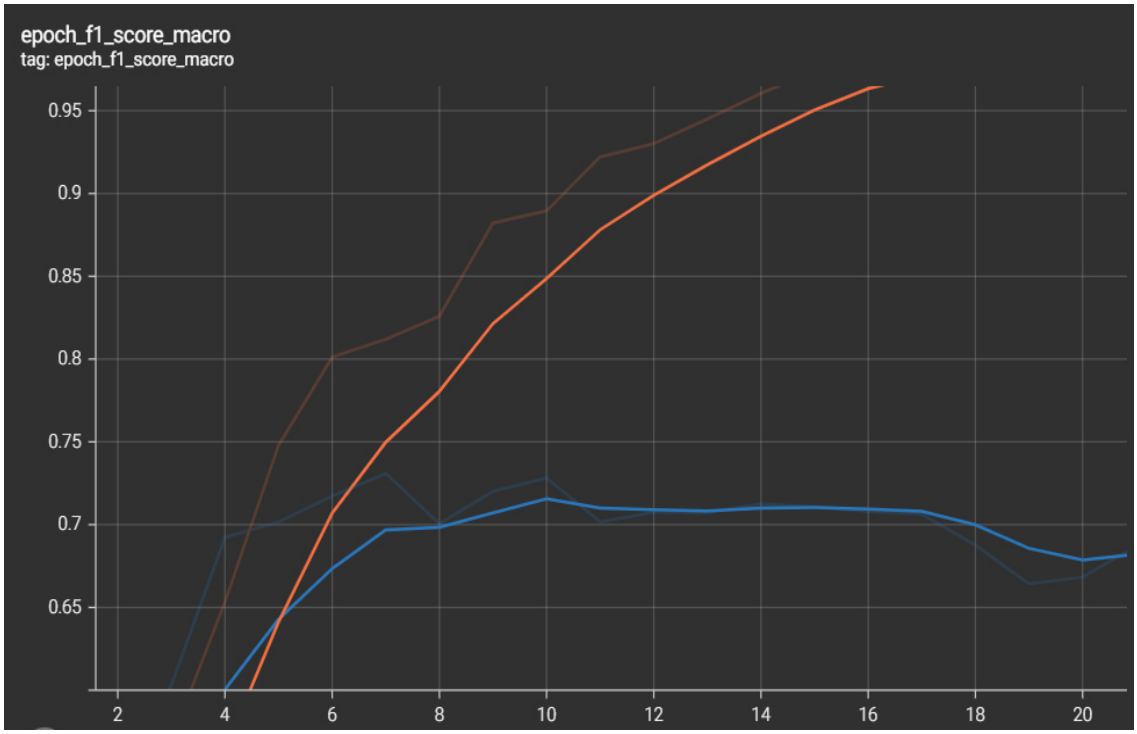


Fuente: Elaboración propia
Nota: La figura muestra la evolución de la curva de entrenamiento y la pérdida (loss) del modelo CNN a lo largo de las épocas. Se observa una disminución progresiva del loss, lo que indica que el modelo mejora su aprendizaje con el entrenamiento. Esto sugiere que el modelo converge adecuadamente y logra un mejor ajuste a los datos.

En la figura 4.15 se observa las curvas de entrenamiento y loss del modelo RNN utilizando un límite máximo de 60 tokens y entrenado a 20 épocas (Early Stopping). La curva azul representa la curva de validation y la curva naranja representa la curva de training. De acuerdo con la gráfica de la curva de validation es que se elegió entrenar el modelo final con 15 epócas.

Figura 4.15 Curvas de entrenamiento y loss del modelo RNN.





Fuente: Elaboración propia
Nota: La figura muestra que el loss de entrenamiento disminuye continuamente, mientras que el loss de validación primero baja y luego aumenta, indicando posible sobreajuste (overfitting). Asimismo, el F1-score de entrenamiento sigue mejorando, pero el de validación se estanca alrededor de 0.70. Esto sugiere que el modelo aprende bien el conjunto de entrenamiento pero no generaliza igual en los datos de validación.

En lo que respecta al mejor modelo para el corpus de tipo variado (Tweet+Debate), la RNN con la capa bidireccional GRU, se observó que configuraciones menores a un límite máximo de tokens de 60 eran altamente inestables con un validation loss que escalaba casi de forma exponencial con el transcurso de las épocas. Al inicio se asumió que usar el límite máximo de tokens encontrado en el corpus (80) sería la mejor opción, ya que la capa de GRU utiliza masking internamente, el cual consiste en identificar las filas que han sido rellenadas por ceros en la secuencia de entrada de modo que estas filas no se utilizen a lo largo del entrenamiento, a diferencia de la CNN por ejemplo en donde no existía un mecanismo así. De la misma forma, al utilizarse el número máximo de tokens como límite el modelo podría utilizar todas las oraciones completas sin truncamientos lo que podría resultar una mejor mezcla.

Sin embargo, esto no fue así, ya que utilizar esta condición para el padding hizo que el modelo se vuelva muy estable mostrando también incrementos casi exponenciales en la curva de validation loss a medida que las épocas iban incrementando. Utilizando un límite de padding relativamente menor (40 tokens) también resultó en una situación similar. Sin embargo, utilizar un límite de padding de 60 tokens dio resultados más positivos en cuanto a estabilidad en las curvas de entrenamiento las cuales se observan en la figura 4.15.

En cuanto a los experimentos realizados con el corpus de solo Twitter y su evaluación en la fuente de datos de debates políticos, según las tablas 4.5 y 4.6, se observó que las arquitecturas basadas en CNN fueron las que mejor resultado obtuvieron. En particular se resalta el comportamiento predictivo de la arquitectura CNN multicapa y la CNN simple en el sentido que con la CNN simple entrenar y evaluar tweets produce un mejor resultado que entrenar con Tweets y evaluar el modelo en datos de escritura y contexto distinto como son los debates electorales. Se concluye que la arquitectura CNN multicapa obtuvo un mayor resultado en la evaluación con los debates políticos ya que esta cuenta con una extracción de características más completa (puede extraer desde bi-grams hasta 4-grams) y esto es beneficioso ante datos con una longitud de token relativamente mayor que el promedio de tokens de todo el corpus (mayor a 30 tokens). El hecho de que la diferencia sea mínima en el score de predicción entre usar un corpus diverso (Tweets + debates) y un corpus de solo tweets abre una puerta a poder entrenar modelos predictivos de clasificación de textos políticos con datos provenientes de diversas fuentes al mismo tiempo y no solo con una sola fuente de datos como se ha estado haciendo en el estado del arte.

En general los resultados se asemejan mucho a los encontrados por (Gavin Abercrombie et al., 2019) usando un dataset relativamente pequeño de 1.6K pseudo-oraciones provenientes de mociones del parlamento del Reino Unido. Aunque la comparación no sea exacta debido a la diferencia en corpus usando para entrenamiento, por la cantidad de datos y el contexto y redacción de los textos políticos que los hace similares sin importar la fuente, por lo que permite afirmar que el modelo alcanza niveles de predicción comparables al estado del arte en clasificación multiclase de textos políticos.

Los modelos de Machine Learning entrenados mostraron un desempeño diferenciado según la técnica utilizada. En particular, el algoritmo Support Vector Machine presentó los mejores resultados en términos de accuracy y F1-score, superando a otros enfoques evaluados.

Este comportamiento puede atribuirse a la capacidad del SVM para manejar espacios de alta dimensionalidad y datos con clases desbalanceadas, características propias del discurso político en redes sociales. Las métricas obtenidas evidencian que el modelo logra una clasificación consistente de los mensajes de proselitismo político, validando su idoneidad para el problema abordado.

4.2. Propuesta de framework para el tratamiento de FakeNews en procesos de elección de autoridades

Al haber propuesto un framework para el procesamiento de lenguaje natural en procesos de elección presidencial, es que se ha visto por conveniente considerar la proposición de un framework adicional que evalue la incorporación de FakeNews en textos políticos. La proposición del FrameWork que atiende las FakeNews está basado en literatura que ha abordado esta problemática. Es preciso manifestar que no se ha hecho experimentación a este respecto, pero podría abordarse en términos prácticos esta problemática.

La metodología solo procesa texto más no imágenes por lo que no comprende modelos multimodales. Asimismo, la predicción solo se basa en predecir un target binario (Noticia Verídica, Noticia Falsa).

La metodología propuesta (preliminar) se detalla a continuación:

4.2.1. Recolección de la fuente de datos

Input: URLs, sitios web, redes sociales.

Output: Dataset de noticias correctamente etiquetadas

Los antecedentes utilizados para esta sección utilizan datasets extraídos de internet, la mayoría utiliza datasets de las elecciones a la presidencia de Estados Unidos 2016, o datasets de benchmarking libres de internet. Es posible hacer una recolección manual de fake news tomando como referencia a lo desarrollado por (Barfar, A, 2022), pero es preciso asegurarse que la forma de etiquetar sea realizado de forma correcta. El resultado de esta etapa es obtener un dataset(s) de texto que contenga noticias tanto falsas como verídicas y que será utilizado para las siguientes etapas de la metodología.

4.2.2. Procesamiento de los datos

Input: Textos raw

Output: Textos limpios listos para ser procesados por un modelo vectorizador

De acuerdo con (Saleh et al., 2021), se propone detalladamente una metodología para la limpieza de texto que comprende los siguientes pasos:

- Conversión a minúsculas
- Remover URLs.
- Remover símbolos especiales
- Remover stop words
- Tokenización
- Stemming

La salida de esta etapa consiste en un dataset que contiene los textos limpios listos para ser procesados por un algoritmo de vectorización. Los pasos a seguir pueden variar según el tipo de dataset recolectado.

4.2.3. Vectorización de los datos

Input: Texto limpio (conjunto de documentos)

Output: Conjunto de vectores que representan a los documentos del corpus (Doc embedding) o un conjunto de matrices donde cada matriz representa un documento vectorizado – cada fila es una palabra del documento.

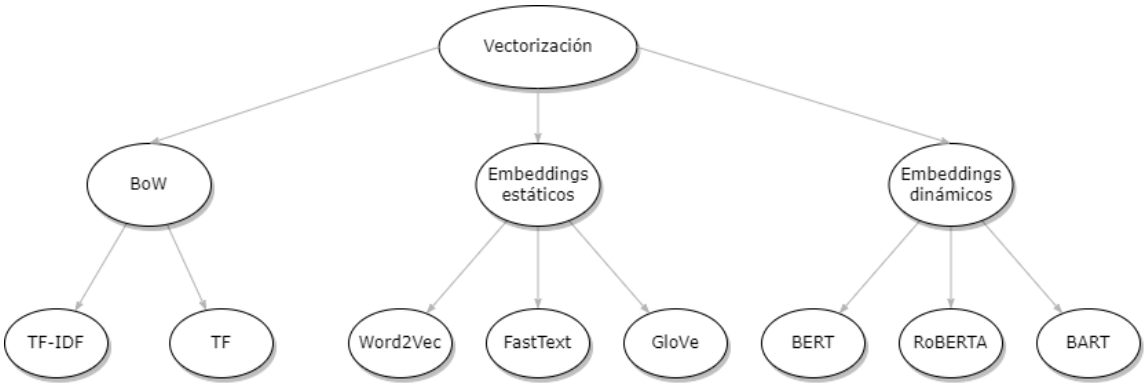
Todos los antecedentes revisados coinciden en que antes de aplicar los algoritmos de predicción es necesario transformar el texto en vectores, ya que los modelos predictivos trabajan con este tipo de dato. Esta metodología divide a la vectorización de datos en 3 grupos: Bag-of-Words, Embeddings estáticos y embeddings dinámicos.

Bag-of-Words (BoW) se refiere al grupo que engloba los enfoques de vectorización por frecuencia como TF-IDF (Term Frequency * Inverse Documento Frecuency) o TF (Term Frequency). (Mayopu et al., 2023) y (Arruda et al., 2020) demostraron que estos enfoques brindan buenos resultados, por lo que también pueden ser considerados como baseline en procesos experimentales.

Embeddings estáticos se refiere al grupo que comprende a los embeddings basados en Word2Vec, entre ellos también se encuentran los modelos FastText y GloVe. (Barfar et al., 2022), analizan los resultados obtenidos usando este tipo de embeddings y concluyen que la elección correcta de estos modelos es crucial para obtener buenos resultados en la predicción. (Saleh et al., 2021) emplean GloVe como capa de embedding para su modelo OPCNN-FAKE. (Arruda et al., 2020) emplearon un conjunto de modelos embedding preetrenados en distintos lenguajes, aunque los resultados no fueron tan satisfactorios comparados a enfoques más clásicos como los basados en BoW.

Embeddings dinámicos se refiere al grupo de embeddings que cambian sus valores según el contexto del texto, los modelos Transformers (BERT, RoBERTA, BART) entran en esta categoría. (Barfar et al., 2022) incluyeron estos embeddings en sus experimentos. Por otro lado, (Kumar, R., 2021) y (Barfar et al., 2022) demostraron resultados satisfactorios al usar BERT como vectorizador y capa de embedding en sus arquitecturas propuestas, encontrando resultados más satisfactorios que modelos comunes de Machine Learning.

Figura 4.16 Modelos de vectorización de texto divididos en grupo.



Fuente elaboración propia
Nota: La imagen muestra las técnicas de vectorización de texto usadas en procesamiento de lenguaje natural. Se dividen en BoW (TF y TF-IDF), embeddings estáticos (Word2Vec, FastText, GloVe) y embeddings dinámicos (BERT, RoBERTa, BART). Esto representa la evolución de métodos para convertir texto en vectores para modelos de IA.

4.2.4. "Liguistic Features"

Input: Texto limpio (conjunto de documentos)

Output: Conjunto de matrices donde cada fila representa a una feature extraída de un documento.

Los investigadores (Barfar, A., 2022) y (Choudhary, A., 2021) emplearon un enfoque distinto a solo vectorizar los datos como en el punto c. Ellos extrajeron características de los textos con los que trabajaron y obtuvieron una lista de variables con las cuales era posible predecir utilizando algún modelo de predicción. Estas características o features pueden ser, por ejemplo, la cantidad de palabras en un documento específico, el score de sentimiento, o dimensiones del lenguaje estándar (como verbos, sustantivos, pronombres) o de tiempo (presente, futuro). (Barfar, 2022) utilizó, por ejemplo, 95 features extraídas con un software especializado. (Arruda et al., 2020) utilizaron features menos complejas (14) donde una de ellas también era el vector obtenido de un modelo embedding. Los resultados obtenidos demuestran que agregar este tipo de features podría mejorar la calidad predictiva del modelo en general.

La metodología plantea el uso de una combinación de las 14 features utilizadas en (Choudhary, A., 2021), donde una de estas features representa también el vector de embeddings del documento. Las 14 features con las que se experimentó están listadas en la tabla 4.9 a continuación:

Tabla 4.9: Feature para el proceso experimental

Id	Feature	Id	Feature
1	Proporción de caracteres en mayúsculas	8	Palabras por frase
2	Proporción de signos de exclamación	9	Proporción de adjetivos
3	Proporción de signos de interrogación	10	Proporción de adverbios
4	El texto tiene signos de exclamación	11	Proporción de sustantivos
5	Número de palabras únicas	12	Sentimiento del mensaje
6	Número de oraciones	13	Representación Embedding
7	Número de caracteres	14	Errores ortográficos

Fuente: (Barfar 2022)
Nota: La tabla presenta 14 características (features) usadas en el proceso experimental para analizar textos. Incluye características lingüísticas y estructurales, como número de palabras, oraciones, uso de signos y proporción de categorías gramaticales. También incorpora sentimiento, embeddings y errores ortográficos para mejorar el análisis del modelo.

4.2.5. Predicción

Input: Matriz de Features extraídas

Output: Score de predicción

Según los antecedentes revisados, hay dos enfoques para hacer las estimaciones: De la forma clásica con Machine Learning y con modelos más complejos utilizando Deep Learning, a continuación, se detallan estos dos enfoques:

Machine Learning:

Esta metodología propone estimadores basados en Machine Learning a modo de baseline, es decir, los experimentos realizados bajo este enfoque servirán de punto de partido para medir si existen mejoras futuras utilizando modelos más complejos de Deep Learning. Deacuerdo con (Truica, 2023), más allá de un modelo complejo, es crucial la elección de un correcto método de vectorización, por lo que aquí se pronoe emplear todos

los mencionados en el punto 4.2.3. Con respecto a los modelos predictivos, se propone emplear los siguientes:

- Support Vector Machine
- Naive Bayes
- Gradiente Boosted Trees
- LightGBM
- AdaBoost
- Logistic Regression

Se propone experimentar con las diversas features propuestas en el punto 4.2.4., así como con los embeddings.

Deep Learning

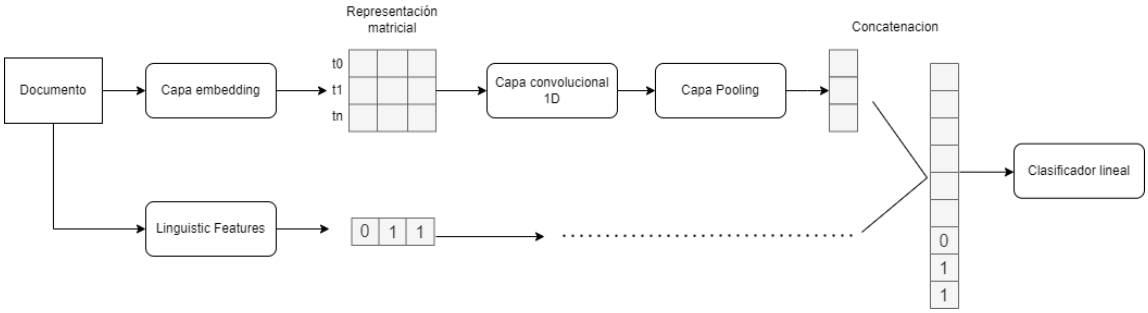
De acuerdo con los antecedentes, (Kumar et al., 2021) y (Saleh et al., 2021) comparten arquitecturas similares basadas en capas convolucionales, los resultados en estos dos artículos son satisfactorios, por lo que para el enfoque de Deep Learning se plantea trabajar con una arquitectura basada en CNNs. A diferencia que con Machine Learning, las capas CNN reciben de input matrices de embedding, por lo que ahora ya no es necesario utilizar doc embeddings bajo algún tipo de promedio.

Por un lado, (Saleh et al., 2021) emplea GloVe como vectorizador, mientras que (Kumar et al., 2021) emplea un enfoque más actual: Transformers con BERT. Esta metodología propone utilizar tanto embeddings estáticos como dinámicos como capa de Embedding en la arquitectura. La diferencia recae en BERT, ya que se pretende trabajar con textos en español, también se empleará la versión española de BERT conocida como BETO (https://github.com/dccuchile/beto) para tener más posibilidades de experimentar con los resultados de predicción.

Por otro lado, también se plantea utilizar redes neuronales secuenciales que toman en cuenta la naturaleza temporal del lenguaje, estas son: LSTM (Long-Short Term Memory) y GRU (Gated Recurrent Unit), asimismo, también se experimentará con la versión bidireccional de estas arquitecturas.

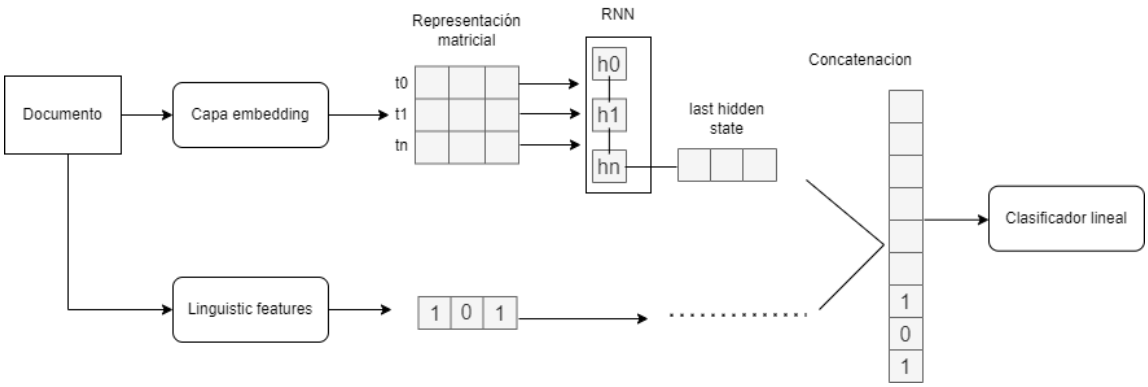
El diagrama de la Figura 4.17 muestra la arquitectura CNN a utilizar.

Figura 4.17 Arquitectura referencial con capas convolucionales



Fuente: Elaboración propia
Nota: La figura muestra una arquitectura de modelo con capas convolucionales para procesar texto. El documento se transforma en embeddings, pasa por una capa convolucional 1D y pooling para extraer características. Luego se concatena con características lingüísticas y se envía a un clasificador lineal para la predicción.

Figura 4.18 Arquitectura referencial con capas recurrentes.



Fuente: Elaboración propia
Nota: La figura presenta una arquitectura con capas recurrentes para el procesamiento de texto. Los embeddings del documento pasan por una capa LSTM y pooling para capturar dependencias secuenciales. Luego se combinan con características lingüísticas y se envían a un clasificador lineal para la predicción.

4.2.6. Métricas de evaluación

Las métricas de evaluación a utilizar para todos los modelos predictivos son las siguientes:

- Accuracy
- Precision

- Recall
- F1-Score
- Curva ROC

Estas métricas permitirán comparar los resultados obtenidos con los de los modelos baseline y también entre cada enfoque, de esta manera se sabrá cual es el mejor modelo.

De manera integral, los resultados obtenidos confirman que el framework propuesto permite articular de forma eficiente las etapas de recolección, procesamiento, análisis y evaluación de información política digital. El desempeño alcanzado por los modelos evidencia que el framework es funcional, reproducible y aplicable al análisis del discurso político en contextos electorales reales.

4.3. Aportes de la Tesis

La principal contribución de la investigación es el Framework para el procesamiento de texto en procesos electorales, fundamentalmente por la combincación de inputs los cuales son:

- Tweets de los principales contendientes en el proceso electoral
- Planes de gobierno
- Transcripción de los debates presidenciales

La propuesta de Framework es inédito en vista que las investigaciones promovidas al interior del Perú al momento de haber iniciado el presente trabajo, solo contemplaban el tratamiento de texto proveniente de redes sociales. El problema de tener como fuente las redes sociales es que para el 2021 el volumen de datos resulta ser insuficiente para procesos de entrenamiento del modelo predictivo.

Por cada una de las etapas se decribe en amplio grado de detalle los pasos seguidos, los que se resumen a continuación:

a. Algoritmo para la extracción de tweets
b. Algoritmo para el análisis y anotación de tweets
c. Algoritmo de extracciónde frases desde fuentes .pdf correspondientes a planes de gobierno

d. Algoritmos para el proceso de limpieza de datos en los tweets planes de gobierno y debates
e. Algoritmos de proceso de vectorización para la experimentación con técncias de ML
f. Algorimto que expone el pipeline para la vectorización
g. Algoritmo de vectorización para experimentos con Deep Learning
h. Algoritmo para el proceso de carga de datos y fine-tunning al modelo words embedding.
i. Algorimo del proceso de entrenamiento, fine-tuning y evaluación de modelos
j. Pipeline para la clasificación de técnicas de Machine Learning
k. Pipeline para la clasificación de técnicas de Deep Learning
l. Algoritmo resumen para la sección de mejor modelo

Respecto a los hallazagos se destaca la capacidad algoritmica del modelo para clasificar los tweets en seis categorías principales, las cuales son:

- Corrupción
- Economía
- Educación
- Pandemia
- Salud
- Seguridad

Así mismo, por cada candidato se ha identificado la distribuación de tópicos de interés.

Respecto al impacto de la investigación emprendida se destacan los siguientes aspectos:

Impacto académico

La tesis tiene varios aportes relevantes para la comunidad científica:

Consolidación de un marco metodológico replicable

El trabajo propone un framework estructurado que integra:

- extracción masiva de datos en redes sociales,
- procesamiento de lenguaje natural,

- clasificación con Machine Learning y Deep Learning,
- evaluación estandarizada (accuracy, F1-score).

Este marco puede ser reutilizado para estudiar:

- otros procesos electorales,
- análisis comparado de discurso político,
- identificación de tópicos en información no estructurada.

Aporte a la literatura latinoamericana

Existe un vacío en estudios técnico–cuantitativos sobre campañas electorales en Perú.

La tesis contribuye con:

- un corpus político propio (tweets, debates, planes de gobierno),
- análisis sistemático del discurso electoral peruano,
- validación de modelos de PLN en español latinoamericano.

Uso de técnicas contemporáneas

El trabajo incorpora:

- Word embeddings,
- SVM optimizado,
- CNN y RNN aplicadas a texto político.

Esto coloca a la investigación en la frontera metodológica para estudios de ciencia política computacional.

Impacto metodológico y tecnológico

Propuesta sólida de pipeline para el análisis automatizado de contenido político

El framework incluye etapas claras:

1. Extracción de datos (Twint, scraping).
2. Limpieza y normalización de texto.
3. Vectorización tradicional y profunda.
4. Entrenamiento y ajuste de modelos.

5. Evaluación estandarizada.

El valor radica en que opera como guía práctica, especialmente útil para:

- equipos de análisis electoral,
- instituciones públicas,
- centros de investigación.

Posible base para sistemas de monitoreo electoral automatizado

El modelo podría extenderse a:

- detección de Fake News,
- análisis de polarización,
- monitoreo en tiempo real de campañas.

Dado que la tesis propone un módulo experimental orientado a tratamiento de Fake News, es factible escalarlo hacia soluciones institucionales.

Impacto institucional (Perú y Latinoamérica)

Aporte a organismos electorales (JNE, ONPE, RENIEC)

Estas instituciones podrían beneficiarse del framework para:

- monitorear narrativas políticas y detectar campañas de desinformación,
- comprender la agenda temática que construyen los partidos,
- mejorar la supervisión del financiamiento digital,
- evaluar cumplimiento de franja electoral.

La tesis aborda explícitamente estas entidades en su contextualización.

Mayor transparencia en campañas digitales

El trabajo evidencia:

- la creciente relevancia de redes sociales desde 2015,
- la vulnerabilidad del sistema ante fake news, bots, influencers políticos,
- la ausencia de marcos formales para estudiar estas dinámicas.

Aplicar este framework permitiría tomar decisiones públicas con mayor base empírica.

Impacto social

Mejor comprensión del rol de redes sociales en elecciones polarizadas

El estudio ayuda a explicar:

- cómo se distribuyen los tópicos políticos,
- qué narrativas predominaron en 2021,
- qué candidatos lograron mayor impacto digital.

Esto disminuye la especulación y aporta datos concretos al debate público.

Contribución al desarrollo de ciudadanía digital

Al visibilizar:

- la vulnerabilidad a noticias falsas,
- la influencia del consumo y replicación de contenido,
- la polarización digital,
- la tesis aporta elementos para campañas de alfabetización digital y formación crítica de votantes.

Potencial para fomentar elecciones más informadas

El framework podría servir para:

- producir reportes imparciales sobre narrativas electorales,
- identificar manipulación digital,
- empoderar a observadores electorales y organizaciones civiles.

Impacto práctico en futuras investigaciones

La tesis abre líneas de investigación como:

- detección de bots políticos en el Perú,
- análisis longitudinal de discursos de gobierno vs campaña,
- comparación entre elecciones 2021–2026,
- integración con datos audiovisuales (TikTok, YouTube).

Además, el corpus creado puede ser base para datasets abiertos de investigación.

4.4. Discusión

Discusión teórica: posicionamiento dentro del campo

La investigación combina teoría política, comunicación digital, inteligencia artificial y análisis del discurso. En este sentido, pone en diálogo disciplinas que usualmente se han estudiado de manera fragmentada.

Redes sociales como actores políticos activos

Los datos muestran que las redes dejaron de ser canales pasivos y se convirtieron en agentes configuradores del discurso, no solo reflejos de preferencias ciudadanas. Esto coincide con teorías contemporáneas sobre:

- agenda setting algorítmica
- cámaras de eco y sesgo de confirmación
- microtargeting político

La tesis contribuye a la discusión al evidenciar cómo las narrativas de 2021 no surgieron únicamente desde partidos o candidatos, sino desde un ecosistema híbrido compuesto por:

- usuarios espontáneos,
- influencers políticos,
- medios alternativos,
- posibles bots,
- cuentas de campaña.

Esto ubica la investigación dentro de la corriente de “política digital no institucional”.

Discusión metodológica: pertinencia y límites del enfoque técnico

La fortaleza central del estudio es su pipeline metodológico, pero también existen desafíos importantes.

Fortalezas metodológicas

Integración de tres fuentes textuales

La combinación de:

- redes sociales (Twitter),
- planes de gobierno,
- transcripciones de debates,

permite una triangulación poco común y muy valiosa. Esto reduce sesgos y da una visión multicapas del discurso político.

Uso de múltiples técnicas de PLN

El empleo comparado de TF-IDF, embeddings, CNN y RNN revela rigurosidad experimental.

Además, la validación con métricas como F1-score macro/micro es coherente con corpus desbalanceados, comunes en política.

Framework replicable

La tesis ofrece una ruta implementable para futuras investigaciones, lo cual es un aporte central.

Limitaciones metodológicas

Sin embargo, es importante reconocer algunos límites:

Dependencia del volumen y representatividad de Twitter

Twitter no representa al votante promedio peruano; suele estar sobrerrepresentado por:

- población urbana,
- sectores politizados,
- jóvenes,
- personas con mayor alfabetización digital.

Aunque esto no invalida el análisis, sí limita su generalización al total del electorado.

Reto semántico: ironía, sarcasmo y ambigüedad

Los modelos tradicionales (SVM, TF-IDF) tienen dificultades para captar:

- ironía política
- sarcasmo
- doble sentido
- referencias culturales peruanas

Los modelos neuronales mejoran esto, pero no lo resuelven completamente.

Problema de labeling manual

El proceso de anotación de tópicos y polaridad, aunque riguroso, siempre está sujeto a:

- subjetividad del codificador,
- variabilidad inter-anotador,
- sesgos cognitivos.

Esto podría ser mitigado con:

- múltiples anotadores,
- coeficiente kappa más estricto,
- validación intersubjetiva.

Discusión contextual: el caso peruano y la excepcionalidad del 2021

La elección peruana de 2021 estuvo marcada por:

- polarización extrema,
- crisis institucional prolongada,
- pandemia,
- desinformación masiva,
- surgimiento de nuevos formatos mediáticos (TikTok, streaming).

La pandemia como acelerador digital

El confinamiento obligó a trasladar la campaña a redes sociales, lo que fortaleció:

- estrategias digitales improvisadas,
- aumento de cuentas anónimas y pseudónimas,
- propagación acelerada de fake news.

La tesis contextualiza correctamente este fenómeno, mostrando que la campaña digital no fue opcional, sino estructural.

Polarización y radicalización

El análisis muestra distribución desigual de tópicos según orientación política.

Esto confirma:

- el voto emocional,
- el discurso negativo (“anti-algo”),
- la fragmentación de agendas temáticas.

El trabajo identifica patrones que coinciden con teorías globales sobre polarización afectiva.

Discusión sobre implicancias prácticas y éticas

Para organismos electorales (JNE, ONPE)

La tesis evidencia la ausencia de:

- regulaciones sobre campañas digitales,
- mecanismos de supervisión de contenido político,
- estrategias contra desinformación sistemática.

El framework propuesto podría convertirse en un sistema de monitoreo automatizado para:

- contenido político,
- anuncios segmentados,
- campañas sucias digitales,
- actividad coordinada no auténtica.

Para partidos políticos

El análisis del discurso revela que:

- la coherencia entre planes de gobierno, redes y debates es baja,
- muchos partidos articulan mensajes inconsistentes según la plataforma,

- los tópicos centrales no siempre reflejan propuestas programáticas.

Esto abre puertas para evaluar accountability discursiva.

Para la sociedad civil

El estudio permite:

- comprender cómo se manipulan narrativas,
- incentivar alfabetización digital,
- visibilizar el rol de influencers noticiosos,
- cuestionar la credibilidad de fuentes informales.

Discusión epistemológica

La tesis se sitúa en la intersección entre:

- epistemología del discurso,
- analítica de datos masivos,
- representación computacional del lenguaje.

El uso de PLN permite una aproximación cuantitativa, pero no puede captar completamente:

- frames ideológicos,
- subtextos culturales,
- narrativas históricas,
- emociones implícitas.

Por ello, se requiere complementar estos modelos con enfoques cualitativos profundos.

Discusión sobre la excepcionalidad del caso Peruano y el número de candidatos

Fragmentación del voto en primera vuelta y su conexión con el análisis digital

La elección presidencial de 2021 contó con 18 postulantes, lo cual generó una dispersión inédita del voto:

- Pedro Castillo: 19 %
- Keiko Fujimori: 13 %

- Terceros y cuartos lugares por debajo del 12 %
- Más del 50 % del electorado repartido entre ocho candidatos

Esta fragmentación tiene implicancias directas en el análisis de redes sociales y PLN desarrollado en la tesis:

Mayor dispersión temática y discursiva en Twitter

La diversidad de candidatos implica:

- múltiples narrativas simultáneas,
- microcomunidades alrededor de propuestas específicas,
- dificultad para identificar un "eje dominante" de discusión.

Los modelos de clasificación de tópicos aplicados en la tesis se vuelven especialmente útiles, ya que permiten agrupar automáticamente discursos atomizados.

Incremento de ruido y señales débiles en los datos

Cuando no existen liderazgos fuertes:

- el volumen de menciones se distribuye irregularmente,
- los picos de atención son episódicos y no sostenidos,
- la huella digital de cada candidatura es más difícil de detectar sin herramientas computacionales.

El framework propuesto permite filtrar ese ruido, haciendo visible lo que a simple inspección sería imperceptible.

La fragmentación electoral refuerza la necesidad de análisis automatizado

La ausencia de candidatos claramente predominantes exige analizar grandes volúmenes de datos textuales para:

- identificar cuáles temas lograron mayor tracción,
- entender qué discursos conectaron con sectores específicos del electorado,

- mapear cómo surgió la figura de Castillo en redes (inusualmente baja al inicio).

Dificultad de identificación con un liderazgo y su impacto discursivo

Cuando el electorado no se reconoce en un liderazgo claro, aparece un patrón que la tesis también detecta en los datos digitales:

Voto emocional disperso

Al no existir una figura dominante, las preferencias se estructuran más por:

- rechazo a determinadas opciones,
- preocupaciones coyunturales,
- ansiedad frente a la crisis sanitaria, económica e institucional.

Esto se refleja en Twitter como:

- mayor volumen de mensajes negativos,
- tendencia al discurso anti–propuesta o anti–candidato,
- radicalización de nichos de apoyo.

Los modelos de diferenciación de tópicos utilizados en la tesis muestran cómo los clusters temáticos son heterogéneos y altamente reactivos, más que programáticos.

El liderazgo débil amplifica la influencia de intermediarios digitales

La ausencia de figuras fuertes genera espacio para:

- influencers políticos,
- periodistas digitales,
- microcreadores de contenido,
- cuentas anónimas o automatizadas.

Esto coincide con estudios citados en la tesis que muestran el peso creciente de actores no institucionales en procesos polarizados.

Las redes sociales reemplazan parcialmente a los liderazgos tradicionales

En contextos con baja identificación partidaria (como Perú), las redes cumplen funciones que antes tenían:

- los partidos,
- los sindicatos,
- los movimientos sociales.

El framework de PLN/Machine Learning permite observar cómo el discurso público deja de estructurarse desde los partidos y se reconfigura desde la "base digital".

Segunda vuelta polarizada: profundización del contraste discursivo

La extrema polarización de la segunda vuelta derivada de que ambos contendientes llegaron con apoyos relativamente bajos pero altamente movilizados tiene implicancias metodológicas y sociopolíticas:

Reconfiguración abrupta de narrativas

El salto de un escenario fragmentado a uno binario implicó:

- colapso de discursos secundarios,
- hiperconcentración temática,
- aumento explosivo del volumen de interacciones.

Los modelos de la tesis logran capturar cómo ciertos tópicos emergen con fuerza, especialmente:

- economía vs. modelo económico,
- comunismo/antagonismos ideológicos,
- corrupción y estabilidad democrática.

Polarización afectiva: las redes como catalizador

Las elecciones de 2021 fueron un ejemplo de manual de polarización afectiva, caracterizada por:

- afecto positivo al propio grupo,
- hostilidad intensa al adversario,
- narrativas moralizantes.

Los resultados de PLN confirman una alta presencia de lenguaje polarizante, lo cual refuerza la idea de que Twitter no solo refleja la polarización, sino que la amplifica.

Mayor dificultad para la ciudadanía no alineada

Gran parte del electorado no se identificaba con Castillo ni Fujimori. Esto produjo:

- incremento de indecisión,
- intensificación del discurso anti–ambos,
- búsqueda de terceros referentes (analistas, influencers).

El análisis computacional evidencia que estos actores intermedios ganaron protagonismo en la conversación digital.

CONCLUSIONES

Conclusión general

El framework metodológico propuesto en la presente investigación trasciende el contexto específico de la elección presidencial peruana de 2021, al haber sido diseñado bajo principios de modularidad, replicabilidad y adaptabilidad. Estas características permiten su aplicación en posteriores procesos electorales, tanto a nivel nacional como subnacional, así como su extensión a otros contextos democráticos con dinámicas comunicacionales similares. La independencia del framework respecto a actores políticos específicos, junto con el uso de técnicas estandarizadas de Procesamiento de Lenguaje Natural y Machine Learning, posibilita su reutilización para el análisis de nuevas campañas electorales, incorporando distintas redes sociales, marcos normativos y coyunturas sociopolíticas, sin comprometer la validez metodológica del modelo

Conclusión específica 1

Se concluye que el framework propuesto constituye el principal aporte metodológico de la tesis, al integrar de manera estructurada técnicas de Procesamiento de Lenguaje Natural y Machine Learning para el análisis conjunto de redes sociales, planes de gobierno y transcripciones de debates presidenciales. Este marco de referencia no solo permitió abordar el proceso electoral peruano de 2021, sino que fue diseñado con criterios de replicabilidad, modularidad y adaptabilidad, lo que lo posiciona como una contribución

aplicable a futuros procesos electorales y a estudios comparativos del discurso político digital.

Conclusión específica 2

En relación con el procesamiento del lenguaje, se concluye que la selección e implementación de técnicas de PLN constituye un aporte técnico-científico, al demostrar su efectividad para transformar datos políticos no estructurados en representaciones analíticas consistentes. La tesis aporta evidencia empírica sobre la pertinencia de los procesos de limpieza, normalización y extracción de características en contextos de alta informalidad discursiva, contribuyendo al cuerpo de conocimiento sobre el análisis automatizado del discurso político en redes sociales y documentos programáticos.

Conclusión específica 3

Se concluye que la aplicación comparativa de técnicas de Machine Learning para el análisis de tópicos políticos representa un aporte analítico y computacional, al validar la capacidad de los modelos supervisados para identificar patrones discursivos relevantes en distintos instrumentos de comunicación política. La tesis demuestra que la integración entre PLN y aprendizaje automático potencia el análisis de grandes volúmenes de datos electorales, ofreciendo un enfoque escalable y reproducible para investigaciones futuras en ciencia política computacional.

Conclusión específica 4

Finalmente, se concluye que la identificación de algoritmos y métricas de evaluación adecuadas constituye un aporte aplicado de la tesis, al evidenciar que modelos como Support Vector Machine, evaluados mediante métricas como accuracy y F1-score, permiten una clasificación confiable de mensajes de proselitismo político en redes sociales. Este aporte valida el uso del framework como una herramienta predictiva útil para el análisis electoral, con potencial aplicación en entidades académicas, organismos electorales y estudios orientados a la detección de patrones discursivos y desinformación.

RECOMENDACIONES

Recomendación 1 – Aplicación y validación del framework en futuros procesos electorales

Se recomienda aplicar el framework metodológico propuesto en futuros procesos electorales, tanto a nivel nacional como subnacional, con el objetivo de validar su replicabilidad y adaptabilidad en distintos contextos políticos y comunicacionales. Esta aplicación permitiría fortalecer su carácter generalizable y consolidarlo como un marco de referencia para el análisis del discurso político digital en escenarios electorales diversos.

Recomendación 2 – Incorporación de técnicas avanzadas de Procesamiento de Lenguaje Natural

Se recomienda extender el framework mediante la incorporación de técnicas avanzadas de Procesamiento de Lenguaje Natural, tales como modelos basados en transformers y embeddings contextuales, que permitan capturar con mayor precisión la semántica, el contexto y la intencionalidad del discurso político. Esta mejora contribuiría a un análisis más profundo de narrativas complejas, ironía y ambigüedad presentes en los mensajes políticos difundidos en redes sociales.

Recomendación 3 – Ampliación del enfoque de aprendizaje automático

Se recomienda explorar enfoques híbridos de aprendizaje automático, combinando técnicas supervisadas y no supervisadas, así como la inclusión de modelos longitudinales

que permitan analizar la evolución temporal del discurso político. Esta ampliación facilitaría la identificación de cambios discursivos, estrategias proselitistas emergentes y patrones recurrentes a lo largo del tiempo electoral.

Recomendación 4 – Extensión del framework hacia la detección de desinformación

Se recomienda extender el framework propuesto hacia la detección y análisis de desinformación y fake news en contextos electorales, incorporando métricas y modelos específicos para la identificación de contenido engañoso. Esta línea de trabajo resulta especialmente relevante para fortalecer la transparencia de los procesos democráticos y apoyar iniciativas académicas e institucionales orientadas al monitoreo del ecosistema digital.

Recomendación 5 – Aplicación práctica en ámbitos académicos e institucionales

Finalmente, se recomienda que el framework sea considerado como una herramienta de apoyo para investigaciones académicas interdisciplinarias y para entidades vinculadas a la gestión electoral, tales como organismos electorales y observatorios de medios digitales. Su uso podría contribuir a la toma de decisiones informadas y al diseño de estrategias de comunicación política basadas en evidencia empírica.

REFERENCIAS BIBLIOGRÁFICAS

Ares, C., & Volkens, A. (2017). ¿Por qué y cómo se está extendiendo el Manifesto Project a América Latina?. *Revista Española de Ciencia Política*

Abercrombie G., et al. (2019). Policy Preference Detection in Parliamentary Debate Motions. *ACL Anthology*

Arruda, P., Ferreira, T. (2020). Fake news detection in multiple platforms and languages. *Expert Systems with Applications*

Barfar, A. (2022). A linguistic/game-theoretic approach to detection/explanation of propaganda. *Expert System with Applications. Elsevier*

Bellman, R. (1978). *An introduction to artificial intelligence: Can computers think?* Boyd & Fraser Publishing Company.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python...* O'Reilly Media.

Bilbao-Jayo, A., et al. (2018). Political discourse classification in social networks using context sensitive convolutional neural networks. *ACL Anthology*

Bodaghy, A. & Oliveira, J. (2020). The characteristics of rumor spreaders on Twitter: A quantitative analysis on real data. *Computer Communications 160 (2020) 674-687*

Bojanowski, P., et al. (2016). Enriching Word Vectors with Subword Information. *https://arxiv.org/abs/1607.04606*

Bravo, F. (2017). Acquiring and Exploiting Lexical Knowledge for Twitter Sentiment Analysis. (Tesis Doctoral). *The University of Waikato, Hamilton, New Zealand.*

Campbell, K. (15 de octubre del 2019). By the numbers: The Strength of ISIS and the Kurds' Fight. *https://www.hstoday.us/subject-matter-areas/counterterrorism/by-the-numbers-the-strength-of-isis-and-the-kurds-fight/*

CCSA (2020). How COVID-19 is changing the world: a statistical perspective. *Creative Commons Attribution 3.0 IGO license.*

Charniak, E., & McDermott, D. (1985). *Introduction to artificial intelligence.* Addison-Wesley.Chatsiou, K., (2020). Text Classification of Manifestos and COVID-19 Press Briefings using BERT and Convolutional Neural Networks. *http://oars.uos.ac.uk/2134/*

Cho et al. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *Cornell University*

Cho et al. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.*

Choudhary, A., Arora, A. (2021). Linguistic feature based learning model for fake news detection and classification. *Expert Systems with Applications*

Ciampiconi, L., et al. (2023). A survey and taxonomy of loss functions in machine learning. *ArXiv [cs.LG]*

Cortes, C. & Vapnik, V. (1995). Support-Vector Networks. *Mach. Learn*

Dey, R., & Salem, M. (2017) Gate-variants of Gated Recurrent Unit (GRU) neural networks. *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA*

Di Cocco, J. & Monechi, B. (2021). How Populist are Parties? Measuring Degrees of Populism in Party Manifestos Using Supervised Machine Learning. *Cambridge University Press*

Elman, J. (1990). Finding Structure in Time. *Cognitive Science, 14: 179-211.*

El Ghamari, M. (2017). Pro-Daesh jihadist propaganda. A study of social media and video games. *Collegium Civitas, Warsaw, Poland*

FastText, (2021). Modelo FastText *https://github.com/dccuchile/spanish-word-embeddings/blob/master/emb-from-suc.md*

Fenoll, V., et al. (2023). Movilización, ecualización y retórica populista en Facebook. Campañas presidenciales de Brasil, Colombia, Chile y Perú. *Revista Comunicación, Universidad de Piura*

Flamino, J., et al. (2021). Shifting Polarization and Twitter News Influencers between two U.S. Presidential Elections. *Cornell University*

Gensim (2021). *https://radimrehurek.com/gensim/intro.html#what-is-gensim*

Github (2021). *https://github.com/pymupdf/PyMuPDF*

Glavas G., et al (2017). Cross-Lingual Classification of Topics in Political Texts. *ACL Anthology*

Gottfried, J., Shearer E. (2016). News Use Across Social Media Platforms 2016. *Pew Research Center*

Grandini, M., et al. (2020). Metrics for Multi-Class Classification: an Overview. *arXiv [stat.ML]*

Gumelar, P. & Aini, N. (2019). Social bot detection on 2019 Indonesia President Candidate's Supporter's Tweets. *Procedia Computer Science 161 (2019) 813 - 820*

Gutmann, M. & Hyvarinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research, 13:307–361*

Gobierno Peruano, (2021). Distribución Financiamiento Público Elecciones Presidenciales 2021 *https://www.gob.pe/institucion/onpe/normas-legales/2518657-rj-1690-2021-jn*

Gobierno Peruano, (2021). Reglamento de Financiamiento y Supervisión de Fondos Partidarios *https://www.gob.pe/institucion/onpe/noticias/1130268-onpe-aprueba-reglamento-de-financiamiento-y-supervision-de-fondos-partidarios*

Gobierno Peruano, (2024). Etapas del proceso electoral. *https://www.gob.pe/institucion/onpe/informes-publicaciones/5371963-etapas-del-proceso-electoral-en-el-peru*

Gonzalez, R. & Woods, E. (2018). Digital image processing. *Pearson.*

Haddi, E. (2015) Sentiment Analysis: Text, Pre-Processing, Reader Views and Cross Domains *(Tesis doctoral) Brunel, University London*

Hallgren, K. (2012). Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial. *National Library of Medicine*

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9(8), 1735–1780*

Hoffman, C. (2020) Artificial Intelligence and knowledge. *CCO 1.0. Universal*

Hunter, J. (2007) Matplotlib: A 2D Graphics Environment, *Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95*

Insigth, (2021). Inclusión acceso a internet *https://www.prysmian.com/en/insight/telecoms/nexst/state-of-broadband-in-peru-ongoing-improvement-and-inclusivity*

Jimenez, L (2019) Definición de un Framework para el Análisis Predictivo de Datos no Estructurados. *Tesis doctoral, Universidad Carlos III de Madrid*

Jurafsky, D. & Martin, J. (2023) Speech and Language Processing *(3rd ed. draft)*

Jones, S. (1972) A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation, Vol. 28 No. 1, pp. 11-21.*

Khatua, A. et al (2020). Predicting political sentiment of voters from twitter in multi-party contexts. *Applied soft computing. Published by Elsevier B.V.*

Kumar, R. et al (2021). FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools and Applications. Springer*

Kendall, E., McGuiness, D. (2019) Ontology Engineering. *Morgan & Claypool Publisher (22)*

Kenter T., et al. (2015). Short text similarity with word embeddings. *ACM Digital Library*

Kim Y., (2014). Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014. *Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar.* Association for Computational Linguistics.

Kingma & Ba, (2014). Adam: A method for stochastic optimization. *Cornell University*

Koh, A., et al. (2021). Predicting Policy Domains from Party Manifestos with BERT and Convolutional Neural Networks. *SocArXiv Paper*

Kurzweil, R. (1990). *The age of intelligent machines.* MIT Press.

Landis & Koch, (1977). The measurement of observer agreement for categorical data. *National Library of Medicine*

Lecun, Y. et al (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324*

Li, L., & Jamieson, K. (2018). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *https://jmlr.org/papers/v18/16-558.html*

Lopez Lopez, et al. (2018). Agenda temática y twitter: elecciones presidenciales en américa latina durante el periodo 2015 -2017. *Profesional de la información*

Luhn, H. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development, vol. 1, no. 4, pp. 309-317*

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*(4), 115–133.

Manifesto (2021) Manifesto project. https://manifesto-project.wzb.eu/

Matplotlib (2021). *https://matplotlib.org/*

Mayopu, R., et al. (2023). Analyzing Online Fake News Using Latent Semantic Analysis: Case of USA Election Campaign. *Big data and cognitive computing*

Mikolov, T., et al. (2010). Recurrent neural network based language model. *INTERSPEECH.*

Mikolov, T., et al. (2013). Efficient Estimation of Word Representations in Vector Space. *Cornell University*

Mikolov, T., et al. (2013). Distributed representations of words and phrases and their compositionality. *In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*

Morin, F. & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. *In Proceedings of the international workshop on artificial intelligence and statistics, pages 246–252, 2005.*

Nilsson, N., (2005) Introduction to Machine Learning. *Stanford University*

Nltk, (2021) *https://www.nltk.org/*

Parkinson, H. (2016). Click and Elect: How Fake News Helped Donald Trump Win a Real Election. *https://www.theguardian.com/commentisfree/2016/nov/14/fake-news-donald-trump-election-alt-right-social-media-tech-companies*

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*

Prechelt, (1998). Early stopping-but when?. *Neural Networks: Tricks of the trade*

Radim, R. & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. *In Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. Valletta, Malta: University of Malta, 2010. p. 46--50. ISBN 2-9517408-6-7.*

Requena, F. (1989). El concepto de red social. *Reis, 48 (49), 137-152*

Rich, E., & Knight, K. (1991). *Artificial intelligence* (2nd ed.). McGraw-Hill.

Russell, S. & Norving, P. (2008), Inteligencia Artificial un enfoque moderno. *Pearson education.*

Saleh, H., et al. (2021). OPCNN-FAKE: Optimized Convolutional Neural Network for Fake News Detection. *IEEE Access*

Seaborn (2021). *https://seaborn.pydata.org/*

Silverman, C. (2016) This Analysis Shows how Fake Election News Stories Outperformed Real News on Facebook. *https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook*

Scikit-Learn (2021). *https://scikit-learn.org/stable/index.html*

Sharma, A. & Ghose, U. (2020). Sentimental Analysis of Twitter Data with respect to General Election in India. *Procedia Computer Science 173 (2020) 325-334*

SUC, (2021). Spanish Unannotated Corpora *https://doi.org/10.5281/zenodo.3247731*

Subramanian, R., et al. (2021). *A survey on Sentiment Analysis*. IEEE Xplore. 11th *International Conference on Cloud Computing, Data Science & Engineering*

SBWCE (2021). Modelo Word2Vec *https://crscardellino.ar/SBWCE/*

Tensorflow (2021). *https://www.tensorflow.org*

Twintproject (2021). *https://github.com/twintproject/twint*

Torres C. (2009) A bíblia do marketing digital. *Editora Novatec. Sao Paulo*

Truica, C., Apostol, E. (2023). It's All in the Embedding! Fake News Detection Using Document Embeddings. *Mathematics.*

Tuttle, L. (2025). ¿Cuál es el mejor lenguaje de programación para IA? *https://www.wgu.edu/blog/what-best-programming-language-ai2503.html*

Waskon, M. (2021). seaborn: statistical data visualization. *Journal of Open Source Software, 6(60), 3021*

Williams, C. (2008). Support Vector Machines. *School of Informatics, University of Edinburgh*

Winston, P. H. (1992). Artificial intelligence (3rd ed.). *Addison-Wesley.*

Wheeler, G., Pereira, L. (2004). *Epistemology and artificial intelligence*. Journal of Applied Logic 2 (4), 469-493

Vinz, S. (14 de octubre del 2015). Developing your theoretical framework. *https://www.scribbr.com/dissertation/theoretical-framework/*

X Sun, D., (2020). Clustering Tweets via Tweet Embeddings. *MIT*

Xu, X., Li, H., Xu, W, Liu, Z., Yao, L and Dai., F (2021). Artificial intelligence for edge service optimization in Internet of Vehicles: A survey. *Tsinghua Science and Technology, vol. 27, no. 2, pp. 270-287*

Zisserman, A. (2015). *C19 Machine Learning* [Course lectures]. University of Oxford.