Nro. 33
JULIO – DICIEMBRE
2025
e-ISSN 2451-5965
Recibido: 05/02/2024
Aceptado: 21/06/2024
Pp.1 - 27
doi.org/10.48162/rev.48.097

Inteligencia Artificial en las Ciencias Sociales: Abordaje desde el Análisis de Sentimientos

Artificial Intelligence in the Social Sciences: Approach from Sentiment Analysis

Inteligência Artificial nas Ciências Sociais: Uma Abordagem a partir da Análise de Sentimentos

Antonio Aguilera Ontiveros
El Colegio de San Luis, A.C
México
antonio.aguilera@colsan.edu.mx

Resumen

La Inteligencia Artificial (IA) se ha desarrollado como un aliado fundamental en las Ciencias Sociales, transformando su ejercicio. En este artículo se alude la importancia de la IA en las Ciencias Sociales, y se ejemplifica a partir de la tarea de investigación de discursos digitalizados. Se presenta el análisis de sentimientos, que puede realizarse con discursos escritos y/o discursos visuales en las redes sociales. Se abordan uno de los modelos matemáticos más característicos de análisis de sentimientos: las máquinas de soporte vectorial (SVM, por sus siglas en inglés) y se programa en Python un algoritmo para interpretar un texto breve usando una SVM. El objetivo principal de trabajo es ser un recurso de referencia fácil, para científicos interesados en el uso de la IA como herramienta metodológica en las Ciencias Sociales

Palabras clave: *IA, Análisis de Sentimientos, Máquinas de Soporte Vectorial, Ciencias Sociales.*

Abstract

Artificial Intelligence (AI) has developed as a fundamental ally in the Social Sciences, transforming their practice. This article alludes to the importance of AI in the Social Sciences and is exemplified by the task of researching digitized discourses. Sentiment analysis is presented, which can be carried out with written discourses and/or visual discourses on social networks. One of the most characteristic mathematical models of sentiment analysis is addressed: support vector machines (SVM), and an algorithm is programmed in Python to interpret a short text using an SVM. The main objective of the work is to be an easy reference resource for scientists interested in the use of AI as a methodological tool in the Social Sciences.

Keywords: *AI, Sentiment Analysis, Vector Support Machines, Social Sciences.*

Resumo

A Inteligência Artificial (IA) tem se desenvolvido como uma aliada fundamental nas Ciências Sociais, transformando a prática da pesquisa. Neste artigo, aborda-se a importância da IA nas Ciências Sociais e exemplifica-se a partir da tarefa de pesquisa de discursos digitalizados. Apresenta-se a análise de sentimentos, que pode ser realizada com discursos escritos e/ou discursos visuais em redes sociais. São abordados um dos modelos matemáticos mais característicos de análise de sentimentos: as máquinas de vetor de suporte (SVM), e programa-se em Python um algoritmo para interpretar um texto breve usando uma SVM. O objetivo principal do trabalho é ser um recurso de referência fácil para cientistas interessados no uso da IA como ferramenta metodológica nas Ciências Sociais.

Palavras-chave: *IA, Análise de Sentimentos, Máquinas de Vetor de Suporte, Ciências Sociais.*

Introducción

La Inteligencia Artificial (IA) ha surgido como una fuerza disruptiva en el ámbito de las Ciencias Sociales, redefiniendo tanto el proceso como el producto de la investigación en este campo multidisciplinario y dinámico. En un mundo cada vez más interconectado y digital, la IA ha impulsado un conjunto de innovaciones que han revolucionado la forma en que los investigadores sociales confrontan los desafíos de las sociedades actuales. Las Ciencias Sociales tienen como su meta principal explicar las sociedades humanas en toda su complejidad. Tradicionalmente, la investigación en este campo se basaba en métodos cualitativos y cuantitativos que involucraban encuestas exhaustivas, entrevistas detalladas, y análisis de datos estadísticos. Si bien estos métodos proporcionaban información valiosa, tenían limitaciones en cuanto a la escalabilidad y a su capacidad para lidiar con la creciente cantidad de datos generados por las sociedades contemporáneas.

Aquí es donde la IA ha revolucionado todo. La capacidad de la IA de procesar grandes conjuntos de datos con una velocidad y escala sin precedentes ha abierto ventanas completamente nuevas sobre cómo se puede hacer investigación en Ciencias Sociales. Los algoritmos de esta tecnología son capaces de identificar patrones, correlaciones y tendencias en conjunto de datos masivos que permiten a los investigadores sondear y comprender fenómenos sociales mucho más en profundidad y con una mayor precisión. Como resultado, se han realizado avances significativos en campos como la sociología analítica, la psicología social y la economía computacional, entre otros, a medida que los investigadores usan la IA para analizar la opinión pública, dinámicas de grupos sociales y comportamientos de consumo, respectivamente.

Un ejemplo concreto que muestra cómo la IA ha impactado la investigación en Ciencias Sociales es el análisis de sentimientos en las redes sociales. La capacidad de esta tecnología de analizar grandes cantidades de datos procedentes de medios sociales en tiempo real, por ejemplo, ha demostrado ser de vital importancia para comprender cómo se forman y cambian las opiniones en línea, las cuales, a su vez, pueden influir sobre decisiones políticas y mercadológicas.

En este trabajo se busca abordar las áreas principales de oportunidad que la IA ofrece a la investigación en las Ciencias Sociales. A manera de explicar el uso de la IA en un área contemporánea de investigación, en este caso en el análisis de discursos digitalizados, se abordará el tema del análisis de sentimientos y las máquinas de soporte vectorial (SVM por sus siglas en inglés) que es la forma más utilizada para llevarlo a cabo, se presentará el modelo matemático y algorítmico de las SVM, así como un ejemplo computacional, usando Python, de cómo llevó a cabo el análisis de sentimientos.

Breve Historia de la Inteligencia Artificial

La Inteligencia Artificial (IA) es un campo fascinante que ha evolucionado durante décadas de avances tecnológicos y descubrimientos científicos. Como resultado, es un campo interdisciplinario que ha influido en múltiples campos, desde la informática hasta la psicología cognitiva. En esta breve historia de la IA, exploraremos algunos hitos importantes en la historia del campo desde su inicio hasta la actualidad.

Se usó por primera vez el término "Inteligencia Artificial" en el *Dartmouth Summer Research Project on Artificial Intelligence* en 1956, que marcó el inicio de la investigación formal en el campo (Kaplan, 2017: 15). Los pioneros de la IA, como John McCarthy, Marvin Minsky y Claude Shannon, llevaron a cabo esta labor con el objetivo de desarrollar programas y máquinas que pudieran realizar tareas que requerían "inteligencia humana", como el razonamiento psicológico y el procesamiento del lenguaje natural.

Durante las décadas de los 50 y 60, la IA se centró en el desarrollo de programas para la resolución de problemas y en lenguajes de programación específicos de la inteligencia artificial. Uno de los primeros programas de IA, el Logic Theorist de Allen Newell, Cliff Shaw y Herbert A. Simon, pudo demostrar teoremas matemáticos automáticamente. El Logic Theorist fue desarrollado en el *Carnegie Institute of Technology* en los Estados Unidos de Norteamérica. Logic Theorist demostró 38 de los primeros 52 teoremas del capítulo dos de los *Principia Mathematica*[1] de Alfred North Whitehead y Bertrand Russell y encontró demostraciones nuevas y más cortas para algunos de los teoremas (Gugerty, 2006).

A medida que la investigación avanzaba, sin embargo, surgieron desafíos significativos. Los enfoques basados en símbolos físicos y la falta de capacidad de cálculo limitaron el progreso. Esto llevó a un período conocido como la “caída de la IA” en los años 1970, en el que el entusiasmo inicial desapareció, debido a unas expectativas poco realistas (cf. Kaplan, 2017: 24-25).

Sin embargo, la IA experimentó un renacimiento en la década de 1980 con la llegada de los sistemas expertos, que eran programas diseñados para imitar la toma de decisiones humanas en dominios específicos. Estos sistemas encontraron aplicaciones en campos como la medicina y la ingeniería (cf. Kaplan, 2017:26-29). La década de 1990 presenció el surgimiento del aprendizaje automático y las redes neuronales artificiales (cf. Vorobioff *et al*., 2022) que son fundamentales para muchas aplicaciones de IA contemporánea. Los algoritmos de aprendizaje automático permiten a las máquinas aprender patrones y así, realizar tareas complejas, desde reconocimiento de voz a clasificación de imágenes. En lo que entrábamos en

[1] Los "Principia Mathematica" de Whitehead y Russell es una monumental obra de matemáticas. Fue escrito por Alfred North Whitehead y Bertrand Russell y publicado en tres volúmenes entre 1910 y 1913 (volúmenes I y II) y en 1913 (volumen III). Tenía como objetivo proporcionar una base lógica y matemática sólida para todas las matemáticas. Los "Principia Mathematica" son ampliamente considerados como uno de los trabajos más influyentes en la historia de la filosofía y las matemáticas del siglo XX.

el siglo XXI, la IA experimentó un renacimiento, impulsado por el acceso a grandes conjuntos de datos y mejoras en el procesamiento de información. Importantes empresas tecnológicas, como Google, Facebook y Amazon, están haciendo cantidad de inversiones en I+D en IA y demostrando aplicaciones prácticas de la misma, como motores de búsqueda mejorados, asistentes virtuales o vehículos autónomos.

Los avances en IA en la última década han sido increíbles, los sistemas de aprendizaje profundo, como las redes neuronales profundas, han superado a métodos tradicionales en tareas NPL (procesamiento de lenguaje natural, por sus siglas en inglés) y de visión por computadora, lo que ha permitido logros como la traducción automática, la detección de fraudes y mejoras en la atención médica.

ChatGPT es uno de los sistemas de procesamiento de lenguaje natural más populares de los últimos tiempos, se basa en la arquitectura GPT (transformadores generativos preentrenados, pro sus siglas en inglés) para integrarse perfectamente con el usuario humano, no importa cuál sea su lengua materna.

OpenAI es una de las empresas más conocidas en procesamiento de lenguaje natural, quienes han desarrollado versión tras versión de ChatGPT, hasta llegar a la actual versión 4. Con ChatGTP un usuario humano puede interactuar directamente desde su lengua natural con el sistema, el que puede recibir instrucciones para hacer básicamente cualquier cosa, desde traducir un texto completo o redactar una receta de cocina, hasta escribir un código en Python o R.

Lo interesante de ChatGPT es le ha permitido al gran público interactuar de forma directa con la Inteligencia Artificial, sin necesidad de ser expertos programadores o científicos en el campo del *Machine Learning*. Otros modelos de IA accesibles al público son Perplexity AI, Claude AI, You AI, Elicit AI, entre muchos más.

¿Qué se entiende por Inteligencia Artificial?

la IA es un campo multidisciplinario que abarca la programación de sistemas y algoritmos que pueden imitar la inteligencia humana en diversas formas. Estas capacidades permiten a las máquinas realizar una amplia gama de tareas, desde el análisis de datos hasta la toma de decisiones, y están en constante evolución a medida que se desarrollan nuevas técnicas y tecnologías en este campo.

Aunque existen diferentes métodos para el aprendizaje de máquinas o *Machine Learning*, los principales modelos de IA se basan en el uso de redes neuronales artificiales (Vorobioff, *et al*. 2022). La siguiente lista enumera diferentes tipos de arquitectura de redes neuronales que se usan comúnmente en la IA:

1. **Redes Neuronales *Feedforward* (FNN):** Estas son las redes neuronales más básicas y ampliamente utilizadas. La información fluye en una dirección, desde la capa de entrada a través de capas ocultas (si las hay) hasta la capa de salida. Se utilizan para problemas de clasificación y regresión (Gurney, 2007).

2. **Redes Neuronales Convolucionales (CNN):** Diseñadas para procesar datos con estructura de cuadrícula, como imágenes y videos. Las capas convolucionales extraen características espaciales y patrones visuales. Se utilizan en tareas de visión por computadora y reconocimiento de imágenes (O'Shea y Nash, 2015).

3. **Redes Neuronales Recurrentes (RNN):** Estas redes están diseñadas para trabajar con datos secuenciales o de series temporales. Tienen conexiones retroalimentadas que les permiten mantener memoria de estados anteriores. Son utilizadas en procesamiento de lenguaje natural (PLN) y tareas de series temporales, como predicción de acciones o traducción automática (Schmidt, 2019).

4. ***Long Short-Term Memory* (LSTM):** Una variante de las RNN que aborda el problema de la desaparición del gradiente. Son especialmente efectivas en tareas de secuencias largas y se utilizan ampliamente en el PLN y el procesamiento de voz (Cheng, *et al*. 2016).

5. **Redes Neuronales Generativas Adversariales (GAN):** Consisten en dos redes, un generador y un discriminador, que compiten entre sí. El generador crea datos falsos y el discriminador intenta distinguir entre datos reales y falsos. Se utilizan para generar contenido nuevo y realista, como imágenes y texto (Goodfellow, *et al*., 2014)

6. **Transformers:** Son una arquitectura de red neuronal que ha revolucionado el procesamiento de lenguaje natural. Se basan en mecanismos de atención y se utilizan en modelos como BERT y GPT para tareas de PLN, como traducción automática, resumen de texto y procesamiento de lenguaje natural en general (Turner, 2023).

7. **Redes Neuronales Autoencoders**: Son utilizadas para tareas de reducción de dimensionalidad y generación de datos. Un autoencoder consta de un codificador que reduce la dimensión de los datos de entrada y un decodificador que los reconstruye. Se utilizan en tareas de compresión de imágenes y detección de anomalías, entre otras (Michelucci, 2022).

8. **Redes Siamesas:** Estas redes se utilizan para comparar dos objetos y determinar si son similares o diferentes. Se aplican en verificación de rostros, reconocimiento de firmas y recomendaciones basadas en similitud (Chen y He, 2020)

9. **Redes Neuronales Residuales (ResNet):** Introducen conexiones de salto para permitir que la información fluya directamente a través de las capas en lugar de solo hacia adelante. Esto ayuda a resolver el problema de desvanecimiento del gradiente y permite entrenar redes más profundas (Wightman, *et al*. 2021).

10. **Redes Neuronales de Memoria a Corto Plazo (MEMN2N):** Diseñadas para tareas de razonamiento y comprensión de texto. Permiten almacenar información en "slots" de memoria para resolver preguntas basadas en el contexto (Sukhbaatar, *et al*., 2015).

Estos son solo algunos ejemplos de las muchas arquitecturas de redes neuronales utilizadas en la IA. La elección de la arquitectura depende del tipo de problema que se desea resolver y del tipo de datos con los que se está trabajando. La IA se beneficia de la diversidad de redes neuronales disponibles para adaptarse a una amplia gama de aplicaciones y desafíos.

La tendencia actual es el uso de la tecnología "*Generative Pre-trained Transformer*" o GTP. GPT es una familia de modelos de lenguaje basados en IA desarrollada por OpenAI. Los modelos GPT son conocidos por su capacidad para generar texto coherente y de alta calidad, y se han utilizado en una variedad de aplicaciones, como generación de texto, traducción automática, resumen de texto y respuesta a preguntas, entre otras.

Otra tecnología es la de Bard de Google, que utiliza en modelos de lenguaje neuronal conversacional, concretamente en LaMDA, desarrollado por la propia compañía. El objetivo es que Bard utilice información de la web con el fin de proporcionar respuestas originales y de calidad. Bard es una IA que usa un modelo de lenguaje diferente a los modelos de lenguaje anteriores, los cuales se centraban en predecir la siguiente palabra en una oración, su modelo LaMDA se enfoca en comprender el contexto y el propósito detrás de las palabras.

Aparte de los métodos de machine learning basados en redes neuronales existen otras aproximaciones matemáticas-computacionales como los siguientes:

- **Árboles de decisión (Decision Trees):** Son modelos que dividen los datos en subconjuntos más pequeños basándose en decisiones lógicas. Son muy útiles para clasificación y regresión. Ejemplo en scikit-learn: *DecisionTreeClassifier*, *DecisionTreeRegressor*. (cf. Izza, *et al*. 2020).

- **Regresión logística (Logistic Regression):** A pesar de su nombre, es un modelo de clasificación que estima probabilidades utilizando una función logística. Ejemplo en scikit-learn: *LogisticRegression* (cf. Islam *et al*. 2024).

- **K-vecinos más cercanos (K-Nearest Neighbors, KNN):** Un método simple pero efectivo que clasifica nuevos puntos basándose en la mayoría de votos de sus k vecinos más cercanos. Ejemplo en scikit-learn: *KNeighborsClassifier (*cf. Cunningham y Delany, 2020).

- **Bosques Aleatorios (Random Forests)**: Una técnica de ensemble que utiliza múltiples árboles de decisión para mejorar la robustez y la precisión del modelo. Ejemplo en scikit-learn: *RandomForestClassifier*, *RandomForestRegressor*. (cf. Louppe, 2015).

- **Boosting (AdaBoost, XGBoost, etc.)**: Métodos de ensemble que combinan múltiples modelos débiles secuencialmente corrigiendo los errores de los modelos anteriores para mejorar el rendimiento. Ejemplo en scikit-learn: AdaBoostClassifier; para XGBoost, se utiliza la biblioteca xgboost. (cf. Florek y Zagdański, 2023).

- **Métodos basados en clustering (como K-Means)**: Utilizados para agrupar datos en k grupos basados en sus características. Los centros de los clusters se ajustan iterativamente. Ejemplo en scikit-learn: KMeans. (Capó, *et al*. 2018)

- **Análisis de componentes principales (Principal Component Analysis, PCA)**: Una técnica de reducción de dimensionalidad que transforma las variables correlacionadas en un conjunto de valores de variables no correlacionadas llamadas componentes principales. Ejemplo en scikit-learn: PCA. (cf. Shlens, 2014).

Estos métodos cubren una amplia gama de aplicaciones y pueden ser adecuados para muchos problemas sin necesidad de recurrir a las redes neuronales, lo que puede ser ventajoso en términos de interpretabilidad y requisitos computacionales.

Otro método muy usado en las aplicaciones de IA son las llamadas *Máquinas de Soporte Vectorial* (SVM, por sus siglas en inglés, *Support Vector Machines*). Las SVM son altamente eficientes en la clasificación binaria y la regresión (Guenther y Schonlau, 2016). Dichos sistemas, han demostrado un rendimiento sólido en una variedad de aplicaciones, incluidas la detección de spam, la clasificación de documentos, el reconocimiento de patrones en imágenes y el análisis de sentimientos en el procesamiento de lenguaje natural. Además, las SVM son capaces de aprender límites de decisión óptimos en un espacio de características de alta dimensionalidad, lo que les permite generalizar bien a datos no vistos. Esto es particularmente valioso cuando se trabaja con datos complejos y no lineales (Moguerza y Muñoz, 2006; Guenther

y Schonlau, 2016). Las características formales-matemáticas y computacionales de las SVM se verán más adelante en este trabajo.

Aplicaciones de la IA en la investigación en Ciencias Sociales

La Inteligencia Artificial (IA) ofrece múltiples aplicaciones en la investigación en las Ciencias Sociales. A continuación, explicaremos cómo se puede utilizar la IA en el trabajo cotidiano del investigador en Ciencias Sociales.

1. **Análisis de Datos Masivos:** La IA es especialmente útil para procesar grandes conjuntos de datos, lo que permite a los investigadores analizar tendencias y patrones en la sociedad. Por ejemplo, se pueden analizar datos de encuestas, registros gubernamentales y redes sociales para comprender mejor el comportamiento y las opiniones de la población.

2. **Análisis de Sentimientos:** La IA puede ser utilizada para el análisis de sentimientos en las redes sociales y otros medios en línea. Esto permite a los investigadores evaluar las opiniones y emociones de la población en tiempo real, lo que es fundamental para el estudio de la opinión pública y la percepción de eventos sociales.

3. **Análisis del discurso digitalizado:** El análisis de los discursos digitalizados es de suma relevancia en el estudio cualitativo en las Ciencias Sociales, el uso de la IA permite el procesamiento de dichos discursos, no importando si son texto o imágenes, generando la extracción de temas o tópicos clave del discurso. Esto implica identificar las palabras o frases más frecuentes y relevantes en el texto que puedan ayudar a entender de qué trata el discurso. Técnicas como el análisis de frecuencia de palabras y el modelado de temas (por ejemplo, usando el algoritmo *Latent Dirichlet Allocation*) se utilizan para este propósito.

4. **Análisis de Redes Sociales:** Los algoritmos de IA pueden identificar tendencias emergentes en las redes sociales al analizar el contenido más popular o mencionado. Esto permite al investigador social estudiar la opinión prevalente o bien el uso de lenguaje de odio, temas relevantes para la sociología digital.

Esta lista es solo demostrativa y no es exhaustiva. Las aplicaciones de la IA son tan grandes que casi cualquier científico social puede beneficiarse de su uso, desde un sociólogo que quiera una lista de referencias con respecto a un tema particular, hasta un politólogo que quiera hacer analizar los millones de comentarios que se vierten en las redes en una campaña presidencial.

Los discursos digitalizados y su relevancia en la investigación social contemporánea

Los discursos digitalizados se refieren a textos, documentos o registros de discursos que han sido convertidos de su forma original en papel o formato oral a un formato digital. Esto implica que han sido escaneados, transcritos o de alguna manera digitalizados para su almacenamiento y análisis en medios electrónicos. Los discursos digitalizados pueden incluir transcripciones de discursos políticos, entrevistas, debates, discursos académicos, conversaciones grabadas, textos literarios, documentos históricos y otros tipos más. También existen los discursos digitalizados que ya son de naturaleza digital desde su origen, como los textos que se encuentran en publicaciones de Facebook o tweets en X. Además, la amplia variedad de contenido digital en sitios web que incluye blogs y podcasts también se considera como discursos digitales.

La importancia actual de los discursos digitalizados radica en la forma en que las sociedades contemporáneas generan datos a una velocidad cada vez mayor y a una escala sin precedentes. Por ejemplo, solo en Facebook hay alrededor de 2,960 millones de usuarios activos mensuales quienes publican aproximadamente 1.93 mil millones de comentarios por segundo[2]. Estos comentarios o publicaciones abarcan tanto texto como video e imágenes. Analizar tal cantidad de datos es una tarea enorme que requiere el uso intensivo de IA para llevar a cabo investigaciones en este ámbito.

El análisis del discurso digitalizado, a menudo conocido como Análisis del Discurso Digital (ADD), es un enfoque interdisciplinario que se enfoca en el estudio sistemático y crítico del lenguaje utilizado en el habla humana en entornos digitales (Cantamutto y Vela-Delfa, 2016). Este campo de estudio ha sido desarrollado en disciplinas como la lingüística, la comunicación, la sociología, la psicología y la ciencia política, entre otras.

El análisis del discurso digital se centra en examinar textos escritos o discursos hablados que se encuentran en formato digital y analizarlos dentro de su contexto completo. Esto implica no solo analizar el contenido textual, sino también tener en cuenta el contexto social.

Se investigan las estructuras lingüísticas presentes en el discurso, incluyendo aspectos como gramática, vocabulario, estilo literario y figuras retóricas. Se presta especial atención a cómo estas estructuras afectan el significado y la capacidad persuasiva del discurso.

El objetivo del análisis del discurso digital es identificar indicadores sociales presentes en el lenguaje, tales como género, clase social, raza, poder y estatus. Se exploran las formas en que el discurso digital refleja y perpetúa relaciones sociales e inequidades tanto dentro de las redes sociales como dentro del mundo digitalizado. Uno de los principales objetivos del análisis del discurso digital es descubrir cómo el lenguaje y el discurso pueden reflejar y perpetuar ideologías, así como distribuciones de poder en una sociedad. Se

[2] Este dato se tomó de https://kinsta.com/es/blog/estadisticas-facebook/

examina cómo se construyen y transmiten discursivamente las normas, los valores y las creencias en los medios digitales.

El análisis del discurso digital a menudo se lleva a cabo desde una perspectiva crítica. Los investigadores cuestionan y critican las estructuras y prácticas de poder presentes en el discurso, buscando revelar sesgos y desigualdades. Esto es extremadamente relevante para generar conocimiento útil para la sociedad, permitiéndole utilizar los medios digitales de manera libre y ética.

Se exploran diversos géneros discursivos, como discursos políticos, discursos científicos, entrevistas, narrativas, medios de comunicación e incluso discursos generados por usuarios comunes en los medios digitales. Cada género tiene sus propias convenciones lingüísticas y retóricas.

Se presta atención tanto a la producción como a la recepción del discurso. Esto implica considerar cómo los hablantes o autores construyen sus mensajes y cómo los oyentes o lectores interpretan y responden al discurso. La ventaja que ofrecen los medios digitales al ser bidireccionales y permitir el diálogo o la controversia es un beneficio que se aprovecha en el análisis del discurso digitalizado. Se busca comprender cómo se da contexto al discurso y cómo se logra la coherencia en el texto en los medios digitales. Esto implica analizar las señales de coherencia tanto a nivel textual como discursivo, así como comprender la naturaleza del entorno digital.

Es importante tener en cuenta que los discursos digitales no se limitan únicamente al texto. Una ventaja de los medios digitales es su capacidad para incluir diferentes formas de comunicación, como voz, video e imágenes. El análisis del discurso digitalizado aborda estos tipos de discursos ampliados utilizando herramientas de IA y enfoques planteados por investigadores como Banks (2010) y Kress y van Leeuwen (2021).

El análisis de sentimientos

El análisis de sentimiento, a menudo descrito como análisis de opinión o minería de opiniones, es una técnica del procesamiento del lenguaje natural (NLP) utilizada para determinar y evaluar la actitud, emoción o sentimiento presente en los textos o un conjunto de datos del texto. El análisis de sentimiento es un proceso automatizado que implica el uso de algoritmos y técnicas de procesamiento de lenguaje natural para evaluar la polaridad emocional o actitud que se expresa en un texto, que puede ser positivo, negativo o neutral. El análisis de sentimiento tiene como objetivo comprender qué piensa o siente una persona o grupo de personas sobre un tema, un producto, un servicio, un acontecimiento o una entidad específica a partir de la información textual disponible. Este proceso se aplica en muchos ámbitos, como la investigación de mercados, la monitorización de redes sociales, la gestión de la reputación en línea, la toma de decisiones empresariales y la comprensión la opinión pública. El análisis de sentimiento utiliza técnicas de procesamiento del lenguaje, aprendizaje automático y modelos de lenguaje

para evaluar con precisión y de forma automatizada la información presente en el texto.

Los métodos computacionales utilizados para el análisis de sentimientos incluyen las llamadas Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés). Las SVM son una técnica de aprendizaje automático comúnmente usada para problemas de clasificación. El análisis de sentimientos puede tratarse como un problema de clasificación en el que se busca que, dada una pieza de texto, se determine si éste lleva consigo un sentimiento positivo, negativo o neutro. A continuación, se describe cómo se emplean las SVM en el análisis de sentimientos:

1. **Preparación de datos**: El primer paso para usar una SVM en un problema de análisis de sentimientos es reunir y preparar un conjunto de datos etiquetado que contenga ejemplos de pedazos de texto junto con sus etiquetas de sentimiento (es decir, positivo, negativo o neutro). Este conjunto de datos se usará para entrenar y evaluar el modelo SVM.

2. **Extracción de características**: Las SVM están diseñadas para trabajar sobre vectores numéricos, por lo que es necesario representar los textos como vectores numéricos. La extracción de características es el proceso de convertir los datos en vectores numéricos. Esto puede incluir el uso de la popular técnica de extracción TF-IDF (Frecuencia de Término-Inversa de Frecuencia de Documento) o el uso de modelos pre-entrenados de lenguaje natural, como Word2Vec[3] o GloVe[4], para representar las palabras como vectores numéricos.

3. **Entrenamiento del modelo SVM**: Después de que los datos se han preprocesado y las características se han extraído, se "entrena" o ajusta el modelo SVM al conjunto de datos con las características y se le entrena usando el conjunto de datos etiquetado. El modelo SVM entrena a través de la búsqueda de un hiperplano óptimo que maximice la distancia entre las diferentes clases de segregación del sentimiento.

[3] En el campo del Procesamiento de Lenguaje Natural (NLP) y el aprendizaje automático, Word2Vec es un modelo de representación de palabras. Fue desarrollado por Tomas Mikolov y su equipo en Google en 2013 y ha sido ampliamente utilizado como una técnica para aprender representaciones vectoriales densas de palabras (también conocidas como *embeddings*) a partir de grandes cantidades de texto no etiquetado. En general, Word2Vec captura la semántica de las palabras representándolas en un espacio vectorial donde palabras similares tienen vectores similares. Al final, Word2Vec genera representaciones vectoriales densas para cada palabra en un vocabulario. También conocido como *embeddings*, estos vectores son de baja dimensionalidad (por ejemplo, 100, 200, o 300 dimensiones es común) que capturan el significado semántico de una palabra y su relación contextual con otras palabras.

[4] GloVe: Global Vectors for Word Representation es otro modelo para la representación de palabras. Desarrollado en la Universidad de Stanford por Pennington, Socher, Manning y publicado en 2014. El modelo GloVe se centra en obtener representaciones vectoriales para palabras. La creación de este modelo se basa en la evidencia de que las palabras con significados similares tienden a aparecer en contextos similares, lo que significa que las palabras están estrechamente relacionadas. Por ejemplo, las palabras " conmocionado " y "asombrado", aunque no son idénticas, a menudo se utilizan en las mismas situaciones, en su mayoría como sinónimos.

4. **Validación y ajuste de hiperparámetros:** Una vez que el modelo se ha entrenado, debe realizarse validación cruzada o división del conjunto de datos para evaluar su rendimiento. Durante esta etapa, se ajustan los hiperparámetros de SVM como el valor de C (que "tunea" el umbral para los errores de clasificación), y el tipo de kernel (lineal, polinómico, RBF, etc.) para optimizar la actuación.

5. **Predicción y evaluación:** Una vez que el modelo se ha entrenado y ajustado, puede ser usado para predecir el sentimiento de textos novedosos. Su actuación se evalúa utilizando las métricas que incluyen la precisión, el recall, y el *f1-score* para determinar qué tan bien este clasificando los textos en las categorías de sentimiento correctas.

6. **Optimización y Ajuste Fino**: Ajustes adicionales en el modelo SVM pueden ser llevados a cabo, tales como la adición de características, o explorando técnicas de procesamiento de lenguaje natural avanzado para obtener mejoras en su rendimiento en el análisis de sentimiento.

7. **Implementado en aplicaciones de Mundo Real**: Este modelo SVM entrenado y validado es implementado entonces en aplicaciones mundo real, tales como sistemas de análisis de redes sociales, chatbots con capacidad de análisis de sentimiento, sistemas de monitoreo de opinión pública, etc.

Las SVM funcionan especialmente bien en análisis de sentimiento porque pueden manejar vectores de características de alta dimensión y descubrir el hiperplano que mejor separa las clases de sentimiento.

Formalismo matemático de las Máquinas de Soporte Vectorial

Las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés, *Support Vector Machines*) son un conjunto de algoritmos de aprendizaje supervisado utilizados en tareas de clasificación y regresión. Su formalismo matemático se basa en encontrar un hiperplano óptimo en un espacio de características que maximice el margen entre diferentes clases de datos. Este formalismo matemático se basa en la optimización de una función de pérdida que penaliza los errores de clasificación y la regularización para controlar la complejidad del modelo. Las SVM son efectivas en tareas de clasificación, especialmente cuando los datos son linealmente separables o se pueden transformar en un espacio de características donde sí lo sean (Moguerza y Muñoz, 2006).

A continuación, se describen los pormenores del formalismo matemático de las SVM (Moguerza y Muñoz, 2006).

En primer lugar, se debe representar los datos. Supongamos que tenemos un conjunto de datos de entrenamiento etiquetado, donde cada

ejemplo se representa como un vector en un espacio de características. Formalmente, tenemos un conjunto de datos (x_i, y_i), donde x_i es el vector de características del ejemplo *i*, e y_i es su etiqueta de clase (1 para la clase positiva o -1 para la clase negativa).

Luego, se debe determinar el objetivo de la SVM. El objetivo principal de una SVM es encontrar un hiperplano en el espacio de características que maximice el margen entre las dos clases. El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase, que se llaman vectores de soporte. Formalmente, el hiperplano se puede representar como $w \cdot x + b = 0$, donde *w* es el vector de pesos y *b* es el término de sesgo.

El tercer paso, es establecer la función de decisión. La función de decisión de una SVM se define como $f(x) = w \cdot x + b$. Para realizar una clasificación, evaluamos esta función de decisión en un nuevo ejemplo *x* y observamos el signo del resultado. Si $f(x) > 0$, clasificamos *x* como perteneciente a la clase positiva; si $f(x) < 0$, clasificamos *x* como perteneciente a la clase negativa.

Después, se hace la optimización del margen. Para encontrar el hiperplano que maximiza el margen, se resuelve un problema de optimización convexa. El objetivo es minimizar la norma del vector de pesos $\|w\|$ sujeto a la restricción de que todos los ejemplos de entrenamiento cumplan con $y_i(w \cdot x_i + b) \geq 1$, donde y_i es la etiqueta de clase del ejemplo *i*. Esta restricción garantiza que todos los ejemplos estén separados del hiperplano por un margen mínimo de 1.

Se establece una función de pérdida. La función de pérdida en una SVM se conoce como la función de bisagra (*hinge loss*), y se define como $L(w, b) = \frac{1}{n}\sum_{i=1}^{n} \left(0{,}1 - y_i(w \cdot x_i + b)\right)$. Esta función penaliza los errores de clasificación y busca minimizarlos.

Por último, se realiza la regularización. En muchos casos, se agrega un término de regularización al problema de optimización para controlar la complejidad del modelo y evitar el sobreajuste. La regularización se suele expresar como $\frac{1}{2}\|w\|^2$, y el objetivo es encontrar un equilibrio entre maximizar el margen y minimizar la norma de *w*.

Un ejemplo de SVM y su uso en el análisis de sentimientos en Python.

En este apartado construiremos dos programas simples en Python[5] para ilustrar la implementación computacional de una SVM y su uso para el análisis de sentimientos. En primer lugar, se realizará el modelo de SVM y se le entrenará. Para la realización de dicho programa se usará la biblioteca *Scikit-Learn* para crear una Máquina de Soporte Vectorial (SVM) para el análisis de

[5] Para un excelente curso introductorio para Python, recomendamos el libro de Eugenia Bahit (2012). Curso: Python para principiantes.

sentimientos. Este programa es un ejemplo básico y utiliza un conjunto de datos de ejemplo para demostrar cómo se puede aplicar SVM al análisis de sentimientos.

El segundo programa, es un ejemplo de aplicación del modelo pre-entrenado al análisis de sentimientos de un texto corto, que consta de solo 335 palabras[6]. Se usaron dos modelos de textos, el primero un texto con tintes negativos y el segundo un texto positivo hacia un producto. El producto que se eligió en el popular juego Geek[7] de Warhammer 40K.

Para correr los programas, es necesario usar un IDE adecuado como lo es Jupyter Notebook, un IDE que funciona sobre su navegador de Internet predeterminado. Se puede también usar el IDLE que viene instalado con la versión que tenga de Python, recomendamos descargar la versión más nueva de Python del sitio oficial[8]. En nuestro caso usamos Python 3.12. Asegúrese de tener Python 3.12[9], Jupyter Notebook, *numpy* y *Scikit-Learn*[10] instalados en su computadora antes de ejecutar los programas. Una vez que tenga instalado Python 3.12 en su computadora, deberá instalar Jupyter[11] Notebook desde el símbolo de sistema en Windows. Este se muestra en la figura 1.

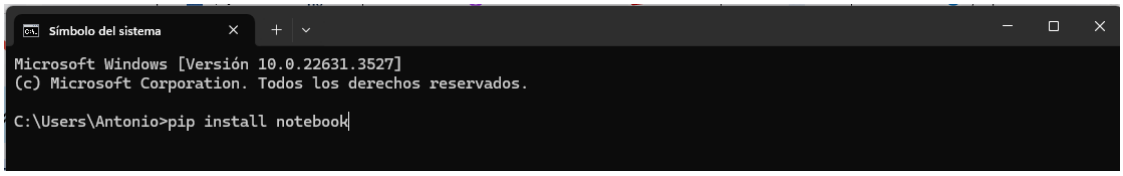

Figura 1. Instalación de Jupyter Notebook.

[6] Un grupo muy grande de palabras, recuérdese que en la longitud máxima de un tweet o post en Twitter es de 280 caracteres. Esto permite escribir aproximadamente de 30 a 40 palabras en un solo tweet, ya que cada palabra tiene un promedio de 5 caracteres. Así que, en términos de palabras, se puede decir que el límite es de alrededor de 40 palabras por tweet.

[7] La cultura geek es un subconjunto de la cultura popular asociada típicamente con temas como la ciencia ficción, la fantasía, los cómics, los videojuegos, la tecnología y la animación japonesa. Algunas características que definen la cultura geek son: a) Afición y conocimiento profundo (a veces obsesivo) sobre temas de nicho como universo de ficción específicos, informática, tecnología, etc; b) Interés en temas que usualmente son vistos como "frikis" o inusuales dentro de la corriente dominante; c) Entusiasmo por coleccionar mercancía como figuras de acción, ediciones especiales, etc. relacionadas con sus temas de interés; d) Asistencia a convenciones y reuniones sobre temas frikis para compartir su pasión con otros; e) Uso de referencias a la ciencia ficción y la fantasía en su vestimenta, accesorios, lenguaje, etc; f) Apego a la tecnología y al uso intensivo de internet y redes sociales; g) Prefieren la fantasía, lo irreal y lo imaginativo versus lo convencional; h) Suelen ser vistos como extraños o inadaptados sociales por quienes no comparten sus intereses.

[8] https://www.python.org/downloads/

[9] Para bajar de la internet Python 3.12 diríjase a la siguiente liga: https://www.python.org/downloads/release/python-3120/

[10] Para aprender de esta biblioteca de machine learning para Python véase: https://scikit-learn.org/stable/

[11] Para mayor referencia véase la página oficial de Jupyter https://jupyter-notebook.readthedocs.io/

Para usar Jupyter Notebook, éste se corre desde el *símbolo de sistema* en Windows o bien *terminal* desde Mac/Linux con el comando siguiente: *jupyter notebook* (ver figura2). Esto abrirá una ventana en su navegador predeterminado donde podrá introducir el código en Python (ver figura 3).

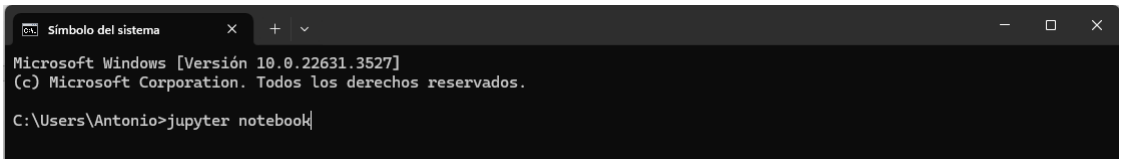

Figura 2. Apertura de Jupyter Notebook.

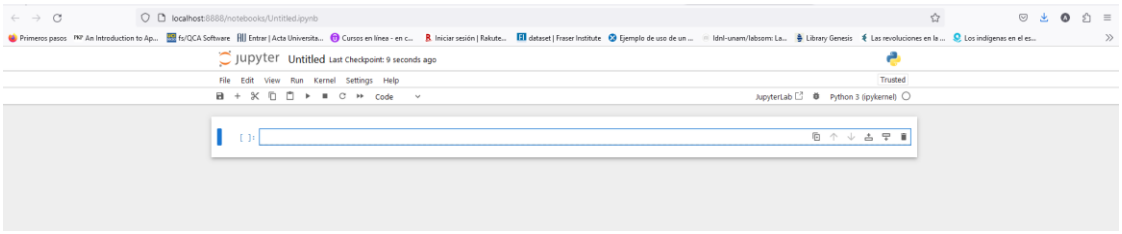

Figura 3. Ventana del IDE Jupyter Notebook en el navegador Firefox.

Como establecimos en un principio, el ejemplo consta de dos programas, El código en python del programa de entrenamiento de una SVM es el siguiente:

```
# Importar las bibliotecas necesarias
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Conjunto de datos de ejemplo

textos = ["Este es un buen producto.", "No me gustó en absoluto.", "Funciona muy bien.",
"No lo recomendaría a nadie.", "Excelente servicio al cliente.", "La peor experiencia que
he tenido.", "No lo volvería a comprar", "creo que es muy caro", "Llegó incompleto", "Horas
de diversión garantizada", "Mi familia lo apreció mucho", "Una gran compra", "una excelente
relación costo-beneficio", "Tiré mi dinero a la basura", "Lo recomiendo ampliamente", "un
asco de producto", "no sé por qué malgasté mi dinero", "no era lo que esperaba", "el mejor
juego que he comprado", "odio este juego", "mejor de lo que esperaba", "las reglas son
complicadas", "lo tuve que devolver", "no podía esperar a comprarlo", "valió la pena la
espera"]

etiquetas = ["positivo", "negativo", "positivo", "negativo", "positivo", "negativo",
"negativo", "negativo", "negativo", "positivo", "positivo", "positivo", "positivo",
"negativo", "positivo", "negativo", "negativo", "negativo", "positivo", "negativo",
"positivo", "negativo", "negativo", "positivo", "positivo"]

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(textos, etiquetas, test_size=0.2,
random_state=42)

# Vectorización de texto utilizando TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)     # Limitar el número de
características para simplificar el ejemplo
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
# Entrenar el modelo SVM
svm_classifier = SVC(kernel='linear')  # Usar un kernel lineal para este ejemplo
svm_classifier.fit(X_train_tfidf, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = svm_classifier.predict(X_test_tfidf)

# Evaluar el rendimiento del modelo
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Precisión: {accuracy}")
print("Informe de clasificación:\n", report)
```

Este código realiza las siguientes acciones:

1. Importa las bibliotecas necesarias, incluyendo *Scikit-Learn* para SVM y la vectorización de texto y *numpy* que significa "Numerical Python," la cual es una biblioteca fundamental en el ecosistema de Python que se utiliza para realizar operaciones numéricas y matemáticas en arreglos y matrices de datos. Recuerde que para instalar dichas bibliotecas se debe hacer desde el símbolo de sistema y con la siguiente sintaxis: *pip install numpy* y *pip install Scikit-Lear*.

2. Define un conjunto de datos de ejemplo que consta de textos y etiquetas de sentimiento (positivo o negativo).

3. Divide el conjunto de datos en conjuntos de entrenamiento y prueba.

4. Utiliza la vectorización de texto TF-IDF para convertir los textos en características numéricas.

5. Entrena un modelo SVM con un *kerne*l lineal en el conjunto de entrenamiento.

6. Realiza predicciones en el conjunto de prueba.

7. Evalúa el rendimiento del modelo utilizando la precisión y un informe de clasificación.

Tenga en cuenta que este es un ejemplo simple con un conjunto de datos pequeño, en nuestro caso solo 25 frases. En aplicaciones reales, se suelen utilizar conjuntos de datos más grandes y técnicas de preprocesamiento de texto más avanzadas para obtener resultados precisos en el análisis de sentimientos.

El resultado de correr este modelo en Python es el siguiente:

```
Precisión: 0.6
Informe de clasificación:
              precision    recall  f1-score   support

    negativo       0.50      1.00      0.67         2
    positivo       1.00      0.33      0.50         3

    accuracy                           0.60         5
   macro avg       0.75      0.67      0.58         5
weighted avg       0.80      0.60      0.57         5
```

Ahora haremos un programa que nos permita usar el programa anterior para interpretar un texto corto y saber el tipo de sentimiento que contiene. Para construir un programa que pueda interpretar un texto corto de alrededor de 400 palabras desde un archivo de texto, primero debemos leer el contenido del archivo de texto y luego aplicar el modelo SVM entrenado para predecir el sentimiento del texto. Se usará las bibliotecas *joblib* y *sys*, la primera será de utilidad para cargar el modelo pre-entrenado desde un archivo, la segunda sirve para cargar el texto a interpretar. Para instalar dichas bibliotecas vaya a símbolo de sistema y use los siguientes códigos (figura 4):

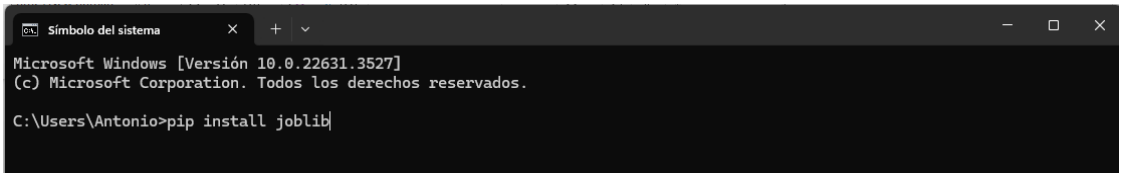

Figura 4. Instalación de la biblioteca joblib

La biblioteca *sys* ya viene instalada por default en Python. Solo la importaremos en el código. El primer paso es guardar en modelo SVM que construimos con el código anterior en un archivo pkl[12], en nuestro caso lo llamaremos “modelo_svm_sentimientos.pkl”

El código es el siguiente, se corre de nuevo el programa inicial pero ahora se sustituye el siguiente código:

```
# Realizar predicciones en el conjunto de prueba
y_pred = svm_classifier.predict(X_test_tfidf)

# Evaluar el rendimiento del modelo
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Precisión: {accuracy}")
print("Informe de clasificación:\n", report)

Por el siguiente código:
```

[12] Los archivos PKL pertenecen principalmente a Python. Un archivo PKL es un objeto serializado creado con el módulo pickle. Contiene cadenas binarias que representan un objeto utilizado en un proyecto Python. Dichos objetos pueden variar desde conjuntos de datos simples hasta modelos de aprendizaje automático.

```
# Guardar el modelo del vectorizador TF-IDF en un archivo
joblib.dump(tfidf_vectorizer, 'vectorizador_tfidf.pkl').

# Guardar el modelo SVM en un archivo .pkl
joblib.dump(svm_classifier, 'modelo_svm_sentimientos.pkl')
```

El nuevo código completo se ve así:

```
# Importar las bibliotecas necesarias
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import joblib  # Para cargar el modelo entrenado desde un archivo

# Conjunto de datos de ejemplo

textos = ["Este es un buen producto.", "No me gustó en absoluto.", "Funciona muy bien.",
          "No lo recomendaría a nadie.", "Excelente servicio al cliente.", "La peor
experiencia que he tenido.",
          "No lo volvería a comprar", "creo que es muy caro", "Llegó incompleto", "Horas
de diversión garantizada",
          "Mi familia lo apreció mucho", "Una gran compra", "una excelente relación
costo-beneficio", "Tiré mi dinero a la basura",
          "Lo recomiendo ampliamente", "un asco de producto", "no sé por qué malgasté
mi dinero", "no era lo que esperaba", "el mejor juego que he comprado",
          "odio este juego", "mejor de lo que esperaba", "las reglas son complicadas.",
"lo tuve que devolver", "no podía esperar a comprarlo",
          "valió la pena la espera"]

etiquetas = ["positivo", "negativo", "positivo", "negativo", "positivo",
             "negativo", "negativo", "negativo", "negativo", "positivo",
             "positivo", "positivo", "positivo", "negativo", "positivo",
             "negativo", "negativo", "negativo", "positivo", "negativo",
             "positivo", "negativo",  "negativo", "positivo", "positivo"]

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(textos, etiquetas, test_size=0.2,
random_state=42)

# Vectorización de texto utilizando TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)    # Limitar el número de
características para simplificar el ejemplo
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Entrenar el modelo SVM
svm_classifier = SVC(kernel='linear')  # Usar un kernel lineal para este ejemplo
svm_classifier.fit(X_train_tfidf, y_train)

# Guardar el modelo del vectorizador TF-IDF en un archivo
joblib.dump(tfidf_vectorizer, 'vectorizador_tfidf.pkl')

# Guardar el modelo SVM en un archivo .pkl
joblib.dump(svm_classifier, 'modelo_svm_sentimientos.pkl')
```

El correr este código nos genera un vectorizador en formato pkl y el modelo SVM en formato pkl. Recordemos que estos formatos son código binarios que usará Python para procesar el texto que introduzcamos y transformarlo a formato vectorial, así mismo se usará el código binario

cargado del modelo SVM para hacer el análisis de sentimientos del texto que introduzcamos.

El programa siguiente es el encargado de procesar el texto que cargaremos en nuestros archivos txt. Se recomienda usar el block de notas de Windows para guardar el archivo de texto que se quiera procesar. El código de nuestro segundo programa quedaría como sigue:

```
# Importar las bibliotecas necesarias
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
import joblib  # Para cargar el modelo entrenado desde un archivo
import sys  # Para leer el archivo de texto

# Cargar el modelo SVM entrenado
svm_classifier = joblib.load('modelo_svm_sentimientos.pkl')  # Asegúrate de tener el
modelo entrenado guardado como "modelo_svm_sentimientos.pkl"

# Cargar el modelo del vectorizador TF-IDF
tfidf_vectorizer = joblib.load('vectorizador_tfidf.pkl')  # Asegúrate de tener el
vectorizador TF-IDF guardado como "vectorizador_tfidf.pkl"

# Leer el texto desde un archivo de texto
archivo_texto = 'WH40Kver01.txt'  # Carga ' WH40Kver01.txt' que contiene el texto
try:
    with open(archivo_texto, 'r', encoding='utf-8') as file:
        texto = file.read()
except FileNotFoundError:
    sys.exit(f"No se encontró el archivo {archivo_texto}")

# Vectorización de texto utilizando el modelo del vectorizador TF-IDF
texto_tfidf = tfidf_vectorizer.transform([texto])

# Realizar la predicción de sentimiento en el texto
sentimiento_predicho = svm_classifier.predict(texto_tfidf)

# Imprimir el resultado de la predicción
if sentimiento_predicho[0] == 'positivo':
    print("El sentimiento del texto es positivo.")
elif sentimiento_predicho[0] == 'negativo':
    print("El sentimiento del texto es negativo.")
else:
    print("El sentimiento del texto es neutral.")
```

Asegúrese de guardar previamente el modelo SVM entrenado como "modelo_svm_sentimientos.pkl" y de que el archivo de texto que desea analizar esté en la misma ubicación o proporciona la ruta completa al archivo en `archivo_texto`. Se recomienda poner en la misma unidad y directorio en dónde se abrió Jupyter Notebook.

Este programa carga el modelo SVM previamente entrenado, lee el contenido del archivo de texto especificado, vectoriza el texto y realiza una predicción de sentimiento. Luego, imprime el resultado de la predicción en función de las etiquetas "positivo", "negativo" o "neutral".

Resultados

A continuación, se especifican los resultados del entrenamiento del modelo de SVM y su interpretación de los dos textos dados sobre la hipotética compra realizada por un fanático de Warhammer 40K.

El primer resultado obtenido son los parámetros de éxito del entrenamiento del modelo SVM. Recordemos que se usó un conjunto de entrenamiento definido por veinticinco frases que corresponden a diferentes opiniones, positivas o negativas con respecto a la compra del producto. Los resultados son los siguientes:

```
Precisión: 0.6
Informe de clasificación:
              precision    recall  f1-score   support

    negativo       0.50      1.00      0.67         2
    positivo       1.00      0.33      0.50         3

    accuracy                           0.60         5
   macro avg       0.75      0.67      0.58         5
weighted avg       0.80      0.60      0.57         5
```

Nótese que la precisión del modelo es adecuada para los fines ilustrativos que queremos realizar en este trabajo. Una precisión mayor se puede lograr incrementando el número de frases en el conjunto de entrenamiento. Sin embargo, veremos que a pesar de tan poca precisión, el modelo es capaz de identificar textos relativamente largos con respecto a si son positivos o negativos.

El texto que se usó para probar el modelo en su vertiente negativa es el siguiente y está guardado en el archivo WH40Kver01.txt.

> No puedo creer que gasté todo ese dinero en el nuevo juego de Warhammer 40K. Qué pésima decisión. Cuando vi el set en la tienda, me dejé llevar por la emoción del momento. Las miniaturas se veían increíbles y la portada prometía horas de diversión estratégica.
>
> Debí haberme dado cuenta que era demasiado bueno para ser verdad. Tan pronto llegué a casa y abrí la caja, supe que había cometido un error. Las instrucciones eran confusas y difícilmente entendibles. Ensamblar las miniaturas resultó ser todo un reto ya que las piezas no encajaban bien. Pasé horas frustrado intentando armar los modelos antes de rendirme.
>
> Decidí buscar algunos videos en línea para tener una mejor idea de cómo jugar. Gran error. Las reglas del juego eran absurdamente complicadas. Había tantas excepciones, fases de movimiento, tiros y dados. Era virtualmente imposible seguir el ritmo. Después de varios intentos fallidos, tiré la toalla.
>
> Este juego fue una pérdida gigantesca de dinero. Por el precio que pagué, esperaba algo divertido y atractivo. En cambio, obtuve un set imposible de armar y un juego imposible de jugar. Mis expectativas se vieron destrozadas por completo. No solo fue una mala compra, sino una estafa total.
>
> Ahora ese costoso juego sólo ocupa espacio en mi closet, recordándome la mala decisión financiera que cometí. Cada vez que lo veo, siento una punzada de arrepentimiento. Podría haber usado ese dinero para comprar varios videojuegos entretenidos o invertir en un pasatiempo más gratificante.
>
> Definitivamente aprendí la lección. La próxima vez no me dejaré cegar por las apariencias llamativas y las promesas grandilocuentes. Haré mi investigación apropiada antes de gastar semejante cantidad de dinero en un producto. Leeré críticas, buscaré opiniones de usuarios reales y me aseguraré de que sea algo que realmente disfrutaré antes de comprarlo.
>
> Por ahora, tengo un costoso recordatorio de no dejarme llevar por la emoción del momento. Ese juego de Warhammer 40K seguirá ahí, burlándose de mi pésimo juicio. La próxima vez seré más cuidadoso e inteligente con mis decisiones de compra.

El resultado de correr el modelo SVM es el siguiente: *El sentimiento del texto es negativo.* Lo cual es correcto al revisar el texto anterior.

El texto que se usó para probar el texto con una tendencia positiva es el siguiente y se encuentra guardado en el archivo WH40Kver02.text.

¡Estoy que no quepo de la emoción con mi nueva adquisición, el juego de Warhammer 40K! Cuando lo vi en la tienda, supe que tenía que hacerlo mío. Las épicas miniaturas de los soldados imperiales y los villanos alienígenas se veían espectaculares. Después de meses ahorrando, finalmente junté el dinero para comprarlo.

Tan pronto llegué a casa, me sumergí de lleno en el maravilloso mundo de Warhammer 40K. Pasé un tiempo increíble ensamblando las decenas de miniaturas detalladas. Cada pequeño soldado era una obra de arte que cuidadosamente pinté con mis colores favoritos. Fue todo un reto, pero muy gratificante ver el fruto final: dos imponentes ejércitos listos para la batalla.

Luego, con gran entusiasmo, me puse a leer el elaborado manual de reglas y vi docenas de videos explicativos en línea hasta entender perfectamente cómo se juega. Debo admitir que las complejas mecánicas tomaron algo de tiempo dominar, pero valió totalmente la pena. Las posibilidades estratégicas eran interminables.

Finalmente, llegó el gran día en que por primera vez enfrenté mis ejércitos en una épica batalla. Fue mejor de lo que esperaba, los dados rodaban y mis soldados avanzaban heroicamente contra las líneas enemigas. Cada choque era más intenso que el anterior con giros dramáticos inesperados. La diversión fue tal que jugamos toda una tarde.

Estoy fascinado con la increíble experiencia que este nuevo juego me ha brindado. Y lo mejor, hay mucho más por explorar. Planeo expandir mis ejércitos, crear nuevos soldados heroicos e incluso construir escenarios personalizados.

No hay duda de que Warhammer 40K ha sido una de las mejores compras que he hecho. No veo la hora de tener mi próxima épica batalla. Invitaré a mis amigos para que también disfruten de esta joya de juego de estrategia. Sin duda, todo fanático de la ciencia ficción debería tener este maravilloso pasatiempo en sus estantes. ¡Lo recomiendo ampliamente!

El resultado de correr el modelo SVM con el texto anterior es el siguiente: El sentimiento del texto es negativo.

Nótese que nuestro modelo no pudo clasificar como positivo el texto anterior. Esto se debe a que mucho del texto, no incluye las frases positivas que hemos utilizado. Veamos ahora qué pasa si incluimos en el texto de tendencia positiva unas tres frases que hemos usado para nuestro conjunto de entrenamiento. Las frases son las siguientes: "*Mi familia lo apreció mucho*", "*el mejor juego que he comprado*", "*una excelente relación costo-beneficio*". El texto se guarda en el archivo WH40Kver03.txt y queda de la siguiente manera y se han subrayado las partes donde aparecen las frases insertadas:

¡Estoy que no quepo de la emoción con mi nueva adquisición, el juego de Warhammer 40K! Cuando lo vi en la tienda, supe que tenía que hacerlo mío. Las épicas miniaturas de los soldados imperiales y los villanos alienígenas se veían espectaculares. Después de meses ahorrando, finalmente junté el dinero para comprarlo. **Mi familia lo apreció mucho**, ya que vieron lo feliz que me hacía. Sin lugar a dudas, este ha sido **el mejor juego que he comprado**.

Tan pronto llegué a casa, me sumergí de lleno en el maravilloso mundo de Warhammer 40K. Pasé un tiempo increíble ensamblando las decenas de miniaturas detalladas. Cada pequeño soldado era una obra de arte que cuidadosamente pinté con mis colores favoritos. Fue todo un reto, pero muy gratificante ver el fruto final: dos imponentes ejércitos listos para la batalla.

Luego, con gran entusiasmo, me puse a leer el elaborado manual de reglas y vi docenas de videos explicativos en línea hasta entender perfectamente cómo se juega. Debo admitir que las complejas mecánicas tomaron algo de tiempo dominar, pero valió totalmente la pena. Las posibilidades estratégicas eran interminables.

Finalmente, llegó el gran día en que por primera vez enfrenté mis ejércitos en una épica batalla. Fue mejor de lo que esperaba, los dados rodaban y mis soldados avanzaban heroicamente contra las líneas enemigas. Cada choque era más intenso que el anterior con giros dramáticos inesperados. La diversión fue tal que jugamos toda una tarde.

Estoy fascinado con la increíble experiencia que este nuevo juego me ha brindado. Y lo mejor, hay mucho más por explorar. Planeo expandir mis ejércitos, crear nuevos soldados heroicos e incluso construir escenarios personalizados.

No hay duda de que Warhammer 40K ha sido una gran compra, con **una excelente relación costo-beneficio**. No veo la hora de tener mi próxima épica batalla. Invitaré a mis amigos para que también disfruten de esta joya de juego de estrategia. Sin duda, todo fanático de la ciencia ficción debería tener este maravilloso pasatiempo en sus estantes. ¡Lo recomiendo ampliamente!

Con esas adecuaciones, el modelo puede identificar al texto como positivo hacia el producto que se compró. Obviamente, hay una estrecha relación entre el conjunto de entrenamiento y los resultados que se esperan de la recolección de datos. Un conjunto *ad hoc* es lo ideal para poder garantizar una adecuada interpretación del texto, a pesar del poco nivel de precisión del modelo, como fue en nuestro caso.

El lector interesado puede generar con el código que se ha descrito su propio modelo de SVM y utilizar sus propios conjuntos de entrenamiento. Entre más amplio sea el espectro de frases adecuadas para identificar los

sentimientos en los datos que se obtienen de fuentes como X o Facebook, mejor nuestro entrenamiento y mejor la predicción del modelo.

Conclusión

En el presente artículo se ha explorado la aplicación de la Inteligencia Artificial (IA), en particular las Máquinas de Soporte Vectorial (SVM), en el contexto de las Ciencias Sociales, centrándose en el análisis de sentimientos como un ejemplo concreto. A lo largo del artículo y el desarrollo de un ejemplo práctico en Python, obramos varias conclusiones clave:

- **Relevancia de la IA en Ciencias Sociales.** La IA se ha convertido en una herramienta esencial en las Ciencias Sociales, proporcionando nuevas formas de abordar problemas complejos relacionados con el comportamiento humano y la opinión pública. El análisis de sentimiento es solo una de las múltiples aplicaciones que la IA tiene en este campo.

- **Aplicación de SVM en el análisis de sentimientos.** En este artículo se ha mostrado que las Máquinas de Soporte Vectorial son eficaces en la tarea de análisis de sentimientos. Su habilidad para hallar hiperplanos óptimos de separación de un espacio multidimensional les permite clasificar textos en categorías de sentimiento de manera precisa.

Un ejemplo muy útil de lo anterior es la implementación de un modelo SVM para un análisis de sentimientos usando Python. En este caso, hemos mostrado cómo la preparación de datos, la extracción de características y el entrenamiento del modelo se combinan para realizar una tarea básica para las Ciencias Sociales. Incluso para quienes ya no dudan de la popularidad del análisis de sentimientos en los datos textuales, o de prácticamente cualquier texto en la Web, seguramente se sorprenderán al enterarse de que la IA se utiliza con asiduidad en la detección de tendencias sociales, en la simulación de comportamiento humano y en el modelado de opinión pública, entre otras prácticas comunes en el quehacer en las Ciencias Sociales.

La versatilidad de la IA y la facilidad con que se encuentra hoy en día es sin duda algo que no tiene precio para la investigación en este campo. Pero, como las buenas noticias nunca vienen solas, es importante señalar que, a medida que avanza la IA en las Ciencias Sociales, se abren nuevos desafíos y oportunidades. Entre ellos se encuentran cómo interpretar los modelos de IA en términos de teorías sociales preexistentes, cómo recoger y usar datos sociales de manera ética y cómo investigar y entender los sesgos algorítmicos.

Como hemos visto, la IA no es un arte oscuro y misterioso reservado para unos pocos iniciados, muy por el contrario, la IA es el producto de modelos matemáticos y computacionales desarrollados durante la segunda mitad del siglo XX y lo que va del XXI y que están disponibles en la Internet para quien quiera usarlos. El uso que le dé un investigador social a este tipo de tecnología para potenciar su investigación e innovar en las Ciencias Sociales dependerá

de qué tan entusiasta y abierto sea ante el cambio de paradigmas en las metodologías de las Ciencias Sociales.

Referencias

Bahit, E. (2012). *Curso: Python para principiantes*. https://archive.org/details/2012CursoPythonParaPrincipiantes

Cantamutto, L., & Vela-Delfa, C. (2016). El discurso digital como objeto de estudio: de la descripción de interfaces a la definición de propiedades. *Aposta, Revista de Ciencias Sociales, 69*, 296-323. https://www.redalyc.org/pdf/4959/495952431011.pdf

Capó, M., Pérez, A., & Lozano, J. A. (2018). An efficient K-means clustering algorithm for massive data. *arXiv*. https://arxiv.org/pdf/1801.02949

Chen, X., & He, K. (2020). Exploring simple Siamese representation learning. *arXiv*. https://arxiv.org/pdf/2011.10566.pdf

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv*. https://arxiv.org/pdf/1601.06733.pdf

Cunningham, P., & Delany, S. J. (2020). *k-Nearest neighbour classifiers: 2nd edition (with Python examples)*. *arXiv*. https://arxiv.org/pdf/2004.04523v2

Florek, P., & Zagdański, A. (2023). Benchmarking state-of-the-art gradient boosting algorithms for classification. *arXiv*. https://arxiv.org/pdf/2305.17094

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *arXiv*. https://arxiv.org/pdf/1406.2661.pdf

Gugerty, L. (2006). Newell and Simon's Logic Theorist: Historical background and impact on cognitive modeling. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 50*(9), 880-884. https://doi.org/10.1177/154193120605000904

Guenther, N., & Schonlau, M. (2016). Support vector machines. *The Stata Journal, 16*(4), 917–937. https://doi.org/10.1177/1536867X1601600

Gurney, K. (2007). Neural networks for perceptual processing: From simulation tools to theories. *Philosophical Transactions: Biological Sciences, 362*(1479), 339–353. http://www.jstor.org/stable/20209847

Islam, R., Mazumdar, S., & Islam, R. (2024). An experiment on feature selection using logistic regression. *arXiv*. https://arxiv.org/pdf/2402.00201

Izza, Y., Ignatiev, A., & Marques-Silva, J. (2020). On explaining decision trees. *arXiv*. https://arxiv.org/pdf/2010.11034

Kaplan, J. (2017). *Inteligencia artificial: Lo que todo el mundo debe saber*. Teell Editorial.

Kress, G., & van Leeuwen, T. (2021). *Reading images: The grammar of visual design*. Routledge.

Louppe, G. (2014). *Understanding random forests: From theory to practice* [Tesis doctoral, University of Liège]. *arXiv*. https://arxiv.org/pdf/1407.7502

Michelucci, U. (2022). An introduction to autoencoders. *arXiv*. https://arxiv.org/pdf/2201.03898.pdf

Moguerza, J. M., & Muñoz, A. (2006). Support vector machines with applications. *Statistical Science, 21*(3), 322–336. http://www.jstor.org/stable/27645765

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv*. https://arxiv.org/pdf/1511.08458.pdf

Schmidt, R. M. (2019). Recurrent neural networks (RNNs): A gentle introduction and overview. *arXiv*. https://arxiv.org/pdf/1912.05911.pdf

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv*. https://arxiv.org/pdf/1404.1100

Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-to-end memory networks. *arXiv*. https://arxiv.org/pdf/1503.08895.pdf

Turner, R. E. (2023). An introduction to transformers. *arXiv*. https://arxiv.org/pdf/2304.10557.pdf

Vorobioff, J., Cerrotta, S., Morel, N. E., & Amadio, A. (2022). *Inteligencia artificial y redes neuronales: Fundamentos, ejercicios y aplicaciones con Python y Matlab*. Editorial de la Universidad Tecnológica Nacional.

Wightman, R., Touvron, H., & Jégou, H. (2021). ResNet strikes back: An improved training procedure in timm. *arXiv*. https://arxiv.org/pdf/2110.00476.pdf