Inteligencia Artificial - Rotten Tomatoes
Universidad Rafael Landívar

Claudia María Chávez Grande
grandeclaudia49@gmail.com
2253524

Angie Paola Schumann Canjura
apschumann8@gmail.com
1201119

Andrea Zucely Raxón Hernández
andreazucely70@gmail.com
1193719

Dulce Maria Fernanda Garcia Díaz
fmarigd@gmail.com
1244621

Lucia Alejandra Cabrera Ordóñez
lucialecabrerao@gmail.com
2510719

Introducción

Este artículo de investigación, desarrollado en el curso de Inteligencia Artificial de la Facultad de Ingeniería de la Universidad Rafael Landívar, busca fortalecer las habilidades en investigación formal en ciencias de la computación. El objetivo es implementar un sistema de clasificación de reseñas de películas de Rotten Tomatoes, determinando si son "Fresh" o "Rotten".

El objetivo es adquirir competencias en búsqueda bibliográfica, pensamiento crítico y argumentación basada en evidencia. Además, desarrollarán habilidades en síntesis y redacción de artículos científicos.

El proyecto se divide en tres entregables: una interfaz gráfica para la carga y normalización de datos, un motor de clasificación en Python, y un artículo de investigación.

Para poder realizar tal proyecto se investigó y estudió diferentes temas, en los cuales fueron el Teorema de Bayes en el cual nos habla sobre poder calcular la probabilidades de un efecto teniendo información de antemano sobre las causas de este, otro tema investigado para la realización del proyecto y uno de los más importantes es “Naive Bayes” o clasificador de reviews en el cual nos da a entender que es un clasificador probabilístico fundamentado en el teorema de Bayes, que suele asumir que las variables predictoras son independientes entre sí.

I. Teorema de Bayes

El Teorema de Bayes, formulado por el reverendo Thomas Bayes, es un principio fundamental en la teoría de probabilidad. Se centra en la actualización de probabilidades basadas en nueva información. El teorema de Bayes es utilizado para calcular la probabilidad de un suceso, teniendo información de antemano sobre ese suceso, dicho principio proporciona un marco para ajustar creencias previas a la luz de evidencia adicional, lo que lo convierte en una herramienta invaluable en el análisis estadístico. [1]

Fórmula

$$P[A_n/B] = \frac{P[B/A_n] \cdot P[A_n]}{\sum P[B/A_i] \cdot P[A_i]}$$

Donde B es el suceso sobre el que tenemos información previa y A(n) son los distintos sucesos condicionados. En la parte del numerador tenemos la probabilidad condicionada, y en la parte de abajo la probabilidad total. En cualquier caso, aunque la fórmula parezca un poco abstracta, es muy sencilla. [2]

II. Probabilidades

Es la parte de las matemáticas que se encarga del estudio de los fenómenos o experimentos aleatorios. Por experimento aleatorio entenderemos todo aquel experimento que cuando se repite bajo las mismas condiciones iniciales, el resultado que se obtiene no siempre es el mismo.

III. Naive Bayes

Es un algoritmo de aprendizaje supervisado que se utiliza principalmente para la clasificación.

Como otros algoritmos de aprendizaje supervisado, se entrena con un conjunto de observaciones que incluye los valores de las ρ variables de entrada y de las categorías de la variable respuesta correspondientes a dichas observaciones. Una vez entrenado el algoritmo, se utiliza para predecir la categoría de la variable respuesta que le corresponde a un conjunto de valores de las variables predictoras.

La primera palabra del algoritmo, Naive, responde al hecho de que se asume que las variables de entrada son independientes entre sí, por lo cual, si se quitan, se introducen o se cambian algunas variables predictoras, los cálculos en los que están involucrados los demás no se ven afectados.

La segunda se debe a que su base es el teorema de Bayes. [6]

IV. Clasificadores Naive Bayes

Es un algoritmo de machine learning supervisado que se utiliza para tareas de clasificación como la clasificación de textos. Utiliza principios de probabilidad para realizar tareas de clasificación.

El clasificador Naive Bayes asume que el efecto de una característica concreta en una clase es independiente de otras características. Por ejemplo, un solicitante de préstamo es deseable o no en función de sus ingresos, su historial previo de préstamos y transacciones, su edad y su ubicación. Aunque estas características sean interdependientes, se siguen considerando de forma independiente. Esta suposición simplifica el cálculo, y por eso se considera ingenua. Este supuesto se denomina independencia condicional de clase. [3]

V. Probabilidades a priori

Se llama iniciales o a priori a las probabilidades $P(A_j)$ de los sucesos que forman las particiones del espacio muestral. [4]

1.1 Implementación de Naive Bayes | A propri

Esta implementación de Naive Bayes en Python utiliza la biblioteca `scikit-learn` para realizar la clasificación supervisada. El proceso comienza importando las bibliotecas necesarias y cargando un conjunto de datos desde un archivo CSV. Los datos se dividen en características (`X`) y etiquetas (`y`), y posteriormente en conjuntos de entrenamiento y prueba utilizando `train_test_split`. Se crea un modelo Gaussian Naive Bayes con `GaussianNB`, que se entrena con los datos de entrenamiento (`X_train`, `y_train`). Luego, se realizan predicciones sobre los datos de prueba (`X_test`) y se evalúa el rendimiento del modelo utilizando métricas como precisión (`accuracy_score`), informe de clasificación (`classification_report`) y matriz de confusión (`confusion_matrix`). Esta metodología proporciona una forma sencilla y eficiente de implementar y evaluar el algoritmo Naive Bayes para tareas de clasificación.

En el contexto del algoritmo Naive Bayes, la probabilidad a priori, P(y), es la probabilidad inicial de cada clase antes de observar cualquier dato. Se calcula como la frecuencia de cada clase en el conjunto de datos de entrenamiento, dividiendo el número de ocurrencias de cada clase por el número total de observaciones. Este cálculo es esencial para determinar la probabilidad general de una clase sin considerar las características, y se utiliza junto con las probabilidades condicionadas de las características para realizar la clasificación. En Python, esto se puede hacer utilizando `np.unique` para contar las frecuencias de las clases y dividiéndolas por el total de muestras, proporcionando así una base inicial para el modelo Naive Bayes.

1.2 Implementación de Fit

La implementación del método `fit` en un modelo Naive Bayes se centra en calcular y almacenar las probabilidades necesarias para realizar predicciones futuras. En el caso de Gaussian Naive Bayes, `fit` implica calcular las probabilidades a priori de cada clase y las estadísticas (media y varianza) de cada característica por clase. Estas estadísticas se utilizan para modelar la distribución gaussiana de las características, asumiendo que siguen una distribución normal. Durante el entrenamiento, el método `fit` toma los datos de entrada (características) y las etiquetas (clases), calcula las frecuencias de las clases para obtener las probabilidades a priori, y luego computa la media y la varianza de cada característica por clase, almacenando estos valores para su uso en el cálculo de las probabilidades a posteriori durante la fase de predicción.

1.3 Entrenamiento de datos

Proceso de entrenamiento y clasificación de reviews. El proceso de entrenamiento consiste en la carga de un archivo txt en formato csv el cual contiene comentarios, debidamente identificados. Se obtiene la probabilidad condicional de cada palabra y la probabilidad de cada etiqueta.Creando así el modelo. Para un comentario no identificado se aplica la fórmula de Bayes para cada etiqueta, esto representa un puntaje que se tomará como referencia para predecir la clasificación.

VI. Laplace Smoothing

El suavizado de Laplace es una técnica de suavizado que maneja el problema de probabilidad cero en Naïve Bayes. Usando el suavizado de Laplace, podemos representar P(w'|positivo) como:

$$P(w'|positive) = \frac{\text{number of reviews with w' and y = positive} + \alpha}{N + \alpha * K}$$

Alfa representa el parámetro de suavizado, **K** representa el número de dimensiones (características) en los datos y **N** representa el número de revisiones con y = positivo.

Si elegimos un valor de alfa no igual a 0, la probabilidad ya no será 0 incluso si una palabra no está presente en el conjunto de datos de entrenamiento. [5]

El suavizado de Laplace es una técnica de suavizado que ayuda a abordar el problema de la probabilidad cero en el algoritmo de aprendizaje automático Naïve Bayes. El uso de valores alfa más altos aumentará la probabilidad hacia un valor de 0,5, es decir, la probabilidad de que una palabra sea igual a 0,5 tanto para las reseñas positivas como para las negativas. Como no obtenemos mucha información de eso, no es preferible. Por lo tanto, se prefiere utilizar alfa=1.

4.1 Posibles problemas de Laplace Smoothing

El suavizado de Laplace, aunque útil para manejar categorías no observadas en Naive Bayes, puede sobresuavizar las probabilidades, haciendo que la distribución sea demasiado uniforme y reduciendo la sensibilidad del modelo a las diferencias reales en las frecuencias de las características. Además, asigna probabilidades mínimas arbitrarias a eventos no observados, lo que puede ser problemático para características poco frecuentes o con significados importantes. Para mitigar estos problemas, se pueden utilizar alternativas como el suavizado de Lidstone, que ajusta una constante menor α en lugar de 1, o enfoques bayesianos con distribuciones a priori más informativas. Ajustar los parámetros de suavizado mediante validación cruzada y considerar modelos específicos del dominio también pueden ayudar a mejorar la precisión y la robustez del modelo.

VII. Implementación

6.1 TfidfVectorizer

Es una biblioteca de scikit-lear, lo que hace es convertir la colección de documentos de texto en una matriz de características Term Frequency-Inverse Document Frequency (TF-IDF), la cual brevemente, es una técnica que refleja la importancia de una palabra en un documento en relación con una colección o corpus de documentos.

¿Qué hace en específico?

1. **Tokenizar:** Divide la cadena reviews en tokens.
1. **Elimina palabras comunes:** Elimina palabras que no aporten mucho significado, como preposiciones.
2. **Construye un vocabulario**: A partir de los documentos, construye un vocabulario.
3. **Cálculo de TF-IDF**: Calcula los valores TF-IDF para cada palabra en cada documento.

6.1 Normalización

Dentro del dataset consideramos NO importantes las columnas 'top_critic', 'review_score', 'review_date', ya que no aportan información o detalle al motor de entrenamiento, haría ruido en nuestra clasificación.

Los modelos de machine learning, como lo es Naive Bayes, requieren que los datos de entrada sean numéricos y en el caso de la columna review_type, se trata de una variable categórica con dos posibles valores: Fresh o Rotten, es por eso que se realiz dicho cambio, siendo:

1: Fresh / 0: Rotten

Por tanto, con lo anterior, se facilita el uso de esta variable como variable objetivo.

6.3 Método trainModel

Extraer las dos columnas del DataFrame y se asignan a una variable.

Fit_transform: este método realiza dos operaciones:

1. **Ajuste (fit):** Aprende el vocabulario y las estadísticas necesarias para calcular el TF-IDF a partir de las reviews en X
2. **Transformación:** Convierte las reviews de texto en una matriz de características numéricas usando el TF-IDF calculado.

El uso de fit_transform simplifica el flujo de trabajo de preparación de datos para modelos de aprendizaje automático al combinar los pasos de ajuste y transformación. Es un método ampliamente utilizado en los pasos de preprocesamiento para garantizar que los datos estén en el formato y escala correctos para el entrenamiento del modelo.

Ajuste: el ‘vectorizer’ analiza todas las reviews en ‘x’ y construye un vocabulario basado en las palabras que aparecen.
Transformación: cada review se convierte en un vector numérico donde cada posición representa el TF-IDF de una palabra del vocabulario.
Esta transformación es esencial para que los algoritmos de machine learning puedan trabajar con datos textuales, convirtiendo el texto en número que pueden ser procesados por el modelo de clasificación.

Se divide el dataset en conjuntos de entrenamiento y prueba.

train_test_split Se utiliza para dividir arrays en dos subconjuntos, uno para entrenamiento y otro para prueba.

Para que el modelo aprenda patrones y relaciones dentro de los datos, necesita una cantidad significativa de datos de entrenamiento, por lo que utilizar el 80% de los datos permite que el modelo tenga una muestra representativa del dataset completo, mejorando así la capacidad del modelo para generalizar a nuevos datos.

Se utiliza el 20% de los datos para pruebas, de esta manera nos permite asegurar que haya suficientes datos para evaluar de manera confiable el rendimiento del modelo, esto es suficiente para obtener una estimación precisa de la precisión, la sensibilidad, la especificidad y otras métricas de rendimiento.

VIII. Conclusiones

La interfaz de usuario debe ser intuitiva y eficiente, permitiendo a los usuarios cargar datos de críticas de películas sin complicaciones. La normalización de los datos es crucial para asegurar que el motor de clasificación opere sobre datos limpios y estructurados, mejorando así la precisión y la eficacia del modelo de clasificación. Los usuarios deben poder crear perfiles y registrar valoraciones de manera sencilla, lo que facilita la recopilación de datos para la clasificación.

El motor de clasificación representa el núcleo técnico del proyecto. Debe ser capaz de procesar críticas de cine y clasificarlas con precisión como "Fresh" o "Rotten". Este motor debe aprovechar técnicas de aprendizaje supervisado y algoritmos de procesamiento de lenguaje natural (NLP) para analizar el texto de las críticas. La integración del motor con la interfaz de usuario debe ser fluida, utilizando REST APIs o invocaciones de procesos para garantizar una comunicación eficaz entre los componentes del sistema.

El artículo de investigación sirve como una documentación detallada del proceso de desarrollo y los resultados obtenidos. Debe incluir una revisión exhaustiva del estado del arte, identificando las técnicas y enfoques más relevantes en la clasificación de textos. La justificación de los algoritmos elegidos debe estar respaldada por literatura científica, y el artículo debe demostrar cómo estos algoritmos se implementaron y evaluaron en el proyecto. Los resultados deben presentarse de manera clara, destacando la precisión y la eficacia del motor de clasificación.

Bibliografía

[1] P. Manero, “Teorema de Bayes: características y su papel en el lanzamiento de productos rumbo a 2024 - Agencia de investigación, estudio e inteligencia de Mercados,” *Investigación de Mercados para el marketing moderno*, Dec. 05, 2023. https://blog.estudiocontar.com/2023/12/05/teorema-de-bayes-caracteristicas/ (accessed May 25, 2024).

[2] J. F. López, “Teorema de Bayes - Qué es, fórmula y ejemplos,” *Economipedia*, Feb. 21, 2018. Accessed: May 25, 2024. [Online]. Available: https://economipedia.com/definiciones/teorema-de-bayes.html

[3] A. A. Awan and A. Navlani, “Tutorial de clasificación Naive Bayes con Scikit-learn,” *DataCamp*, Mar. 13, 2024. Accessed: May 25, 2024. [Online]. Available: https://www.datacamp.com/es/tutorial/naive-bayes-scikit-learn

[4] “¿Qué es el Teorema de Bayes y cómo se aplica?,” *StudySmarter ES*. https://www.studysmarter.es/resumenes/matematicas/estadistica-y-probabilidad/teorema-de-bayes/ (accessed May 25, 2024).

[5] V. Jayaswal, “Laplace smoothing in Naïve Bayes algorithm,” *Towards Data Science*, Nov. 22, 2020. Accessed: May 25, 2024. [Online]. Available: https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece

[6] *Capítulo 27 Naive Bayes*. [Online]. Available: https://cdr-book.github.io/cap-naive-bayes.html