

ISIS 3301: Inteligencia de Negocios

Proyecto 1: NLP

Tema escogido: Revisión de sentimientos en comentarios de libros

Integrantes del Grupo

* María Alejandra Pabón | 201728807 | ma.pabon
* Melissa Castañeda | 201731437 | mc.castanedap
* Juan Diego Sánchez | 201823001 | jd.sancheza1

Octubre, 2021

**Proyecto 1: Revisión de sentimientos en comentarios de libros**

**Contenido**

**1. Introducción**

**2. Comprensión del negocio y enfoque analítico.**

**3. Comprensión de los datos y preparación de los datos.**

**4. Modelado y evaluación.**

**5. Resultados (Tablero de control + estrategias)**

***DESARROLLO***

1. **Introducción**

En este proyecto trabajamos con los datos de Amazon reviews: Kindle Store Category, el cual es un pequeño subconjunto del conjunto de datos de reseñas de productos de la categoría de Amazon Kindle Store. El dataset mostrado contiene alrededor de 1 millón de comentarios sobre más de 61 mil libros en la red de Goodreads, con la utilidad de los reviews.

El propósito de este proyecto sería analizar los textos para identificar qué tan útil podría ser un review de un libro para quienes lo leen. Con esta problemática, se considera que la tarea de clasificación nos ayuda a plantear correctamente una solución.

Por lo tanto en este documento se explicará el proceso que realizamos para realizar la minería de texto y los modelos de clasificación haciendo uso de las técnicas: KNN por Melissa Castañeda, árboles de decisión por Juan Diego Sanchez y random Forest por Alejandra Pabon. El objetivo principal de un problema de clasificación es identificar la categoría/clase a la que pertenecerán los nuevos datos, en este caso la variable objetivo es helpful la cual muestra que tan buena fue la “review” dada.

1. **Comprensión del negocio y enfoque analitico**

| Oportunidad/problema Negocio | Como fue mencionado anteriormente, el propósito de este proyecto será analizar los textos para poder predecir qué tan útil podría ser un review de un libro para quienes lo leen. Y con esto identificar cuántos y cuáles son los libros con los reviews más útiles. Esto nos permitirá hacer un análisis de cuáles son los libros con reviews más populares. De esta manera se podría ayudar con la organización de las reviews, priorizando las que puedan ayudar más a los usuarios y así lograr ofrecer un mejor servicio recomendando libros. |
| --- | --- |
| Descripción del requerimiento desde el punto de vista de aprendizaje de máquina | Desde el punto de vista de aprendizaje de máquina se busca que por medio de la analitica de textos encontremos los reviews que son más útiles y con la ayuda de los algoritmos de clasificación. De esta manera se pueden categorizar las diferentes reseñas según su utilidad, la cual va a depender de el texto dentro de la reseña. |

**Detalles de la actividad de minería de datos**

| Tarea | Técnica | Algoritmo e hiper-parámetros utilizados (con la justificación respectiva) |
| --- | --- | --- |
| Clasificación | Árbol de decisión | DecisionTree(). Se escogieron estos valores de hiperparámetros ya que, por medio de GridSearch, se obtuvo que éstos hiperparámetros generaban el valor más alto de precisión. |
| Clasificación | KNN | knn(). Se escogieron estos valores de hiperparámetros ya que, por medio de GridSearch, se obtuvo que éstos hiperparámetros generaban el valor más alto de precisión. |
| Clasificación | Random Forest | randomForest(). Se escogieron estos valores de hiperparámetros ya que, por medio de GridSearch, se obtuvo que éstos hiperparámetros generaban el valor más alto de precisión. |
| Preparación de datos | Tokenización | A las palabras de los reviews se les aplicó tokenización: cada reseña es una lista y sus elementos son las palabras. Estas palabras se pasaron a mayúsculas, se cambiaron los números a texto, se eliminaron los signos de puntuación, se eliminaron los caracteres que no son ASCII y se eliminaron palabras vacías. |
| Normalización de datos | Lematización | Para la normalización, se hizo uso de lematización cuyo parámetro fueron las palabras. La lematización consiste en transformar una palabra que está en su forma flexionada a su lema original. |

1. **Los datos y preprocesamiento**

Usamos un conjunto de 1 millon datos de de comentarios sobre más de 61 mil libros, las columnas son:

* asin - ID of the product, like B000FA64PK
* helpful - helpfulness rating of the review - example: 2/3.
* overall - rating of the product.
* reviewText - text of the review (heading).
* reviewTime - time of the review (raw).
* reviewerID - ID of the reviewer, like A3SPTOKDG7WBLN
* reviewerName - name of the reviewer.
* summary - summary of the review (description).
* unixReviewTime - unix timestamp.

Se cargaron 15000 filas de datos, ya que todo el documento pesaba demasiado y generaba demasiadas demoras al procesar los datos. Además, 15000 datos son una buena muestra del conjunto total. También, se eliminaron las columnas que no nos proporcionan información para el reconocimiento de qué tan útil es una reseña, las columnas eliminadas fueron: *asin, overall, reviewTime, rewiewerID, reviewerName, unixReviewTime.*

Se rectificó de que todos los datos de las columnas fueran de tipo string para poder hacer el correcto perfilamiento de datos. Seguido a esto se obtuvo cierta información sobre cada fila, como el conteo de palabras, la moda, la palabra más larga y la más corta. La figura 1 ilustra una muestra de esta información.

 Imagen 1.

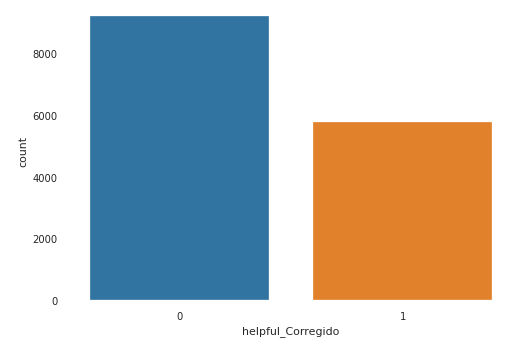
Como se trabajan con textos escritos por personas en internet, éstos no siempre están bien escritos, por lo que hay que tener un proceso detallado para limpiar estos datos. Con la librería de Contractions, se aplica la corrección que brinda esta a los datos para corregir las contracciones que puedan existir (you´re -> you are). Luego, a cada dato se le aplicó una tokenización (dejar las palabras en una lista) y corrección de palabras, la cual incluye pasar las palabras a minúscula, los números (2) se pasan a palabras (dos), se eliminan los signos de puntuación, se eliminan todos los caracteres que no estén dentro de ASCII y finalmente se eliminan las palabras vacías (como artículos, pronombres y preposiciones).

Para normalizar los datos, se aplica entonces lematización y eliminación de prefijos y sufijos. La lematización consiste en transformar una palabra que está en su forma flexionada a su lema original, por ejemplo: “dijo” -> “decir”.

Después de la normalización, se vuelven a unir las palabras que estaban en el arreglo en una sola oración, para posteriormente hacer la matriz correspondiente que nos ayudará con la tarea de clasificación.

Como se mencionó anteriormente en el documento, la variable objetivo en nuestro caso será “Helpful”, la cual nos indica qué tan útil es una reseña. Como indica el diccionario de datos, la variable “helpful” está en formato entre corchetes ([2,3]), lo cual indicaría que fue de una ayuda de 2/3. Con esta información, se procede a modificar esta columna de tal manera que se haga la división de el valor del arreglo en la primera posición sobre el valor del arreglo en la segunda posición, este valor se reemplaza en su fila respectiva y al final se tendrá un valor entre 0 y 1.

Para poder identificar cuando una reseña es útil y cuando no, se crea una clase la cual convierte el valor de “helpful” a 0 o 1; si el valor resultante de helpful es menor a 0.5, la reseña no resulta tan útil entonces este valor se transforma a 0; si por el contrario el valor de la columna helpful es mayor a 0.5, la reseña se considera útil por lo que su valor se transforma a 1. Esto nos ayuda a identificar cuándo una reseña es útil y cuándo no y así poder tener una variable objetivo binaria que nos ayude a resolver el problema del negocio. La figura 2 muestra la distribución de los datos con este valor

 Imagen 2.

Para la representación final, usaremos la matriz de conteo, es una matriz donde cada columna representa una palabra del diccionario final y cada fila una reseña. Cada celda tiene el número de veces que aparece la palabra correspondiente en la reseña.

1. **Modelado y evaluación**

Para la tarea de Clasificación que ayuda con la solución del problema del negocio, se escogieron tres diferentes algoritmos: Árboles de decisión, a cargo de Juan Diego Sánchez, Vecinos más cercanos KNN a cargo de Melissa Castañeda y Random Forest a cargo de María Alejandra Pabón. A continuación se muestra la explicación e implementación de cada algoritmo

* Árboles de decisión:

Los árboles de decisión son un tipo de modelo de clasificación supervisada de clasificación. Estos se basan en dividir los datos por ramas por según atributos donde, cada nodo que es una una rama representa una clasificación sobre un atributo y las hojas son las decisiones de predicación sobre una variable objetivo. Esto funciona separando los datos por particiones hasta que cada instancia de datos pueda ser clasificada con una hoja.

Los hiperparámetros más importantes de los árboles de decisión son el criterion que determina cómo se hacen los splits y el número máximo de splits. Aunque el módulo de DecisionTreeClassifier de Sklearn, con el cual lo modelamos en este ejercicio, tiene otros hiperparametros, estos dos mencionados son los más importantes. Los otros son para ser más especificos con el árbol y le permiten al modelador tener más flexibilidad sobre los splits y el modelo en general.

Para el caso, lo que hicimos fue armar un árbol de decisión con la librería sklearn. Entonces cuando lo armamos optimizamos los híper parámetros de forma exhaustiva con K Fold partitions cross validation para encontrar el óptimo.

* KNN:

KNN es un algoritmo de aprendizaje perezoso y no paramétrico. No paramétrico significa que no hay suposiciones para la distribución de datos subyacentes, esto quiere decir, que la estructura del modelo se determina a partir del conjunto de datos.

Se llama algoritmo perezoso porque no necesita ningún punto de datos de entrenamiento para la generación del modelo. Todos los datos de entrenamiento se utilizan en la fase de prueba, lo que hace que el entrenamiento sea más rápido y que la fase de prueba sea más lenta.

El vecino más cercano K (K-NN) es un algoritmo simple que almacena todos los casos disponibles y clasifica los nuevos datos o casos en función de una medida de similitud.

Para lograr ejecutar un modelo de KNN debemos dividir los datos en entrenamiento y test en donde se encuentra la variable objetivo que dijimos que era helpful. Con esta información podemos sacar que tan exactos son los datos de test y los de entrenamiento.

Un factor que es importante resaltar en el algoritmo KNN, son los hiperparámetros, éstos son el número de n vecinos a inspeccionar, este es especificado cuando creamos el modelo. Y el parámetro p es p de Minkowski fórmula. Cuando p se establece en 1, esto equivale a usar manhattan\_distance, y euclidean\_distance se usará si a p se le asigna el valor 2. En nuestro caso p es igual a 2 y el número de vecinos es 7.

* Random Forest:

El modelo de Random Forest es un modelo de clasificación el cual hace uso de árboles de decisión: es la unión de diferentes árboles de decisión individuales. La manera en la que el algoritmo crea los diferentes árboles es por medio de “bootstrapping”, el cual es un método estadístico que usa el muestreo aleatorio con reemplazos para modificar los datos.

Este método, al hacer uso de múltiples árboles de decisión, posee muchas de las características de estos que se mencionaron anteriormente. Sin embargo, una cita de “The elements of statistical learning” menciona que los árboles de decisión poseen un aspecto que les impide ser ideales, el cual se relaciona con su no muy alta precisión. Es por eso que Random Forest combina la simplicidad de los árboles de decisión con la flexibilidad, resultando en un mejoramiento en la precisión. Para su funcionamiento, tenemos lo siguiente:

* Seleccionamos nuestro conjunto de entrenamiento, el cual debe ser aleatorio, y lo separamos del conjunto de test.
* Luego, se comienza creando el primer árbol de decisión haciendo uso de atributos de los datos escogidos con cierta aleatoriedad, para que entre árboles varíe la toma de decisión. Sin embargo, por cada atributos escogidos aleatoriamente, el algoritmo escogerá el que genere un mejor rendimiento.
* Seguido a esto, se crean más árboles haciendo el mismo recorrido que el paso anterior, para así tener varios árboles distintos entre sí pero que también nos ayuden a solucionar el problema de manera correcta. La variedad de árboles es lo que hace a Random Forest más efectivos que los árboles de decisión individuales.
* Una vez se tienen los árboles de decisión, haciendo uso del conjunto de entrenamiento se entrena entonces el modelo.
* Finalmente, haciendo uso del conjunto de test, se prueba el modelo. Para cada paciente, se le preguntará a todos los árboles del Forest si este presenta diabetes o no. Dependiendo de la respuesta mayoritaria, se escogerá ésta como predicción para dicho paciente.

Otro factor que es importante resaltar en el random forest, son los hiperparámetros que afectarán su rendimiento. Cómo está compuesto de varios árboles de decisión, los hiperparámetros de este entrarán en juego: la profundidad del árbol. En el random forest, la profundidad del árbol será igual para todos los árboles. Otro hiperparámetro importante es la cantidad de árboles: pocos árboles no generarán una buena precisión y sería lo mismo que tener un solo árbol de decisión; muchos árboles generarán que el algoritmo se aprenda perfectamente el conjunto de entrenamiento pero no resulte en buenos resultados para el conjunto de test.

Para el desarrollo de este proyecto, se realizó una búsqueda de hiperparámetros por GridSearch, una herramienta que nos brinda sklearn para buscar los mejores hiperparámetros para nuestro modelo. Al ser tantos los datos que se deben procesar, se sabía que la profundidad de los árboles y la cantidad de árboles debía ser grande. Al terminar la búsqueda de hiperparámetros, se obtuvieron unos valores de: max\_depth = 100 y n\_estimators = 200, además de un criterio de entropía para la búsqueda.

1. **Resultados**

| **Modelo** | **Exactitud** | **Recall** | **Precisión** |
| --- | --- | --- | --- |
| **Árbol de Decisión** | 63% | 30% | 56% |
| **KNN** | 63% | 6,6% | 62% |
| **Random Forest** | 68% | 32% | 68% |

Con base a las métricas obtenidas, se puede observar que el mejor algoritmo que se puede implementar sería el de Random forest, ya que también posee la exactitud más alta. Se observa también que el recall en todos los algoritmos no es muy alto, lo que traduce que el modelo genera varios Falsos Negativos, es decir que varias reseñas que deberían ser clasificadas como “útiles”, se clasifican como “no útiles”. Se observa que para todas las métricas, también, Random Forest es el que predomina en este aspecto.

**REPOSITORIO:** https://github.com/ale1201/Proyecto1BI