Reconocimiento de Patrones y Aprendizaje Automatizado

Clasificador de Documentos

M. CONCHA VÁZQUEZ
L. HERNÁNDEZ CANO
M. X. LEZAMA HERNÁNDEZ
E. E. VÁZQUEZ SALCEDO

Facultad de Ciencias, UNAM.

Facultad de Ciencias
Universidad Nacional Autónoma de México
http://www.fciencias.unam.mx/

Título del proyecto:
Sistema de clasificación automática de críticas de películas.

Tema del proyecto:
Aprendizaje Supervisado; Clasificación de Documentos (de texto).

Periodo:
Febrero-marzo, 2018.

Número de Grupo:
7085

Participantes:
Concha Vázquez Miguel.
Hernández Cano Leonardo.
Lezama Hernández María Ximena.
Vázquez Salcedo Eduardo Eder.

Supervisores:
Profesor Gustavo De la Cruz Martínez.
Rafael Robles Ríos.

Fecha en que se completó el proyecto:
2 de marzo del 2018.

Resumen:

Aplicamos técnicas de aprendizaje automatizado en la clasificación de reseñas de películas basada en la calificación numérica que los críticos asignan a las películas. En particular empleamos las implementaciones del algoritmo C4.5 y las Redes Bayesianas en el software *Weka* como clasificadores para varios conjuntos de datos y analizamos y comparamos los resultados obtenidos.

Índice general

Capítulo 1

Planteamiento

1.1. Introducción

La industria cinematográfica es uno de los negocios más grandes de la economía moderna. De acuerdo con el informe más reciente de Statista, las proyecciones en el aumento de ingresos que genera se estima se incrementarán hasta alcanzar los cincuenta mil millones de dólares anuales para el año 2020. Aunado a ésto, la facilidad para la población de consultar reseñas de estas películas ha crecido enormemente con el auge del Internet, surgiendo sitios como Letterboxd, IMDb, Rotten Tomatoes, entre otros. La cantidad de críticos y opiniones publicadas ha visto un incremento masivo en las últimas décadas. Sin embargo, a la par cada vez hay más usuarios y aficionados que escriben reseñas de películas a partir de análisis que en ocasiones podrían ser catálogados como más superficiales y sin adentrarse demasiado en los aspectos técnicos del cine. Se vuelve relevante entonces analizar la relación entre el contenido de la crítica de una película y el número que asigna el crítico a la película, con tal de poder calificarla de forma automática a partir de lo que logra plasmar el individuo con su recensión.

1.2. Definición del problema

Para un tratamiento formal del problema definamos algunos conceptos básicos.

Definición 1 (Crítica) *Un crítica c es una pareja (t, n), con $0 \leq n < k$, k el numero de clases para la clasificación y t una cadena del idioma (lenguaje) inglés. El valor de n es* ***la calificación*** *de c, y t es* ***el contenido*** *de c. Las críticas también serán llamadas ejemplares.*

La restricción del idioma es necesaria por las técnicas que usaremos para tratar el problema, sin embargo su elección particular es motivada solamente por la disponibilidad de los datos que usaremos. El valor de k sí tiene consecuencias, en nuestro problema trataremos el caso particular cuando $k = 2$ (clasificación *booleana* o *binaria*) pues es un modelo apropiado de la forma en la que las críticas están calificadas, logrando plasmar una polaridad de la misma, pudiendo significar que la apreciación de la película fue positiva o negativa.

Notemos que un conjunto de críticas puede separarse en el subconjunto de aquellas críticas cuya calificación es uno, el subconjunto de aquellas cuya calificación es 2, y así hasta k. Esto nos lleva a definir una clasificación.

Definición 2 (Clasificación de un conjunto de críticas) *La clasificación de un conjunto de críticas C es una partición $C = [C_1, ..., C_k]$ tal que para toda $c \in C_i$, la calificación de c es $i - 1$.*

Definición 3 *Decimos que i es la clasificación de c si $c \in C_i$.*

Definición 4 *Decimos que $1, ..., k$ son las clases de C.*

Lo cuál nos permite definir un elemento fundamental de nuestro problema.

Definición 5 (Clasificador de críticas) *Un clasificador de películas es un proceso que dado un conjunto de críticas C, genera una partición $C^* = [C_1^*, ..., C_k^*]$. Decimos que un elemento $c \in C_i^*$ está mal clasificado en C^* si $c \notin C_i$.*

La gran diversidad y la complejidad lingüística de los lenguajes naturales hacen la tarea de diseñar un proceso que clasifique inviable con las herramientas algorítmicas desarrolladas a día de hoy. Debido a la dificultad de escribir un programa convencional que realice la tarea de clasificador, nos interesa entonces una técnica que clasifique con un error razonable. Por lo mencionado no podemos indicar de forma explícita cada regla *a priori*, lo que nos lleva a considerar un proceso que aprenda de la experiencia y tenga un desempeño razonable. Definimos estos conceptos.

Definición 6 (Medida de desempeño) *La función P de evaluación del desempeño de un clasificador de críticas está definida como sigue. Si C es un conjunto de críticas, $C_1, ..., C_k$ es su clasificación y $C_1^*, ..., C_k^*$ es la particición generada por un clasificador de críticas, la imagen de P bajo C^* es la razón el entre número de elementos de C que no están mal clasificados en C^* y el número total de elementos de C.*

Definición 7 (Aprendizaje de un programa) *Decimos que un programa de computadora aprende de la experiencia E con respecto a alguna tarea T y medida de desempeño P, si su desempeño en T con respecto a P mejora con la experiencia E [***bell***].*

En nuestro caso la experiencia E sería un conjunto de críticas recopiladas al que más tarde llamaremos *Conjunto de Entrenamiento*, P sería la función de desempeño definida y la tarea T sería clasificar críticas de otro conjunto de críticas sin calificación, al que más tarde llamaremos *Conjunto de Prueba*.

El problema entonces es encontrar un proceso que cumpla la definición 7 con lo especificado. De acuerdo con esto nos interesan todas las funciones que reciban un conjunto de críticas y devuelvan una partición, y para navegar este espacio utilizaremos un conjunto de críticas etiquetadas con su calificación. Esto motiva las siguientes definiciones.

Definición 8 (Espacio de hipótesis) *Una función que aproxima a la función objetivo es llamada una* ***hipótesis****. Al conjunto de funciones que el proceso que realiza un aprendizaje puede considerar como candidatos se le llama* ***espacio de hipótesis****. La función objetivo es llamada también hipótesis verdadera.*

El espacio de hipótesis de nuestro problema es el conjunto de funciones que reciben el cuerpo de una crítica y cuya imagen es una calificación.

Definición 9 (Aprendizaje supervisado (1)) *Cuando un proceso observa parejas entrada-salida y aprende una función que aproxima el comportamiento observado en esos ejemplos, se trata de un caso de aprendizaje supervisado [***russell***][p. 695]. La función que produjo los datos de ejemplo es llamada función objetivo.*

Definición 10 (Aprendizaje supervisado (2)) *Dado un conjunto de entrenamiento de N parejas etiquetadas con una entrada y la salida esperada $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, en donde cada y_i, con $i \in \{1, \ldots, N\}$, es generada por una función desconocida $y = f(x)$, el aprendizaje supervisado trata de aproximar la función por medio de una hipótesis h [***russell***][p. 695].*

De acuerdo con esta definición el proceso que realizaremos para obtener el clasificador de críticas es un ejemplo de aprendizaje supervisado.

Definición 11 (Capacidad de generalizar) *Cuando una hipótesis obtiene la misma imagen que obtendría la hipótesis verdadera para un ejemplar nuevo, decimos que* ***generaliza*** *apropiadamente [***russell***][p. 696].*

Definición 12 (Sobreajuste) *Cuando una hipótesis que describe a un conjunto particular de datos es inecesariamente compleja, se dice que hay un sobreajuste [***oxfordstatistics***]. Esto puede resultar en una capacidad para generalizar baja.*

La etiqueta de una crítica puede ser su calificación, por lo que el aprendizaje del proceso es un ejemplo de aprendizaje supervisado.

Capítulo 2

Metodología

2.1. Descripción de la técnica

Las técnicas que emplearemos son los *Árboles de decisión* y las *Redes Bayesianas*. A continuación ofrecemos un modelo de nuestros datos que se ajusta al trato con estas técnicas.

Definición 13 *Dado un conjunto de críticas C, los atributos de C son todas las palabras que pertenecen al contenido de alguna crítica en C.*

Definición 14 *El valor de un atributo w para c es el número de veces que aparece w en el cuerpo de c.*

Definición 15 *El valor binario de un atributo w para c es uno si w aparece en el cuerpo de c y cero en otro caso.*

2.1.1. Árboles de decisión

La primera técnica nos ofrece un marco teórico para representar a los procesos clasificadores como árboles de decisión.

Definición 16 (Árboles de decisión) *Un árbol de decisión para un conjunto de críticas C es un árbol finito dirigido en el que cada nodo terminal está etiquetado con una clase de C, cada nodo interno está etiquetado con un atributos de C y las aristas que salen de un nodo w son los valores posibles de w en los elementos de C (adaptación de [***moshkov***]).*

Tenemos una definición recursiva para la imagen del proceso clasificador representado por el árbol de decisión.

Definición 17 *La clasificación de una crítica c bajo un nodo terminal etiquetado con la clase c^*, es la misma clase c^*. En otro caso, la clasificación de una crítica c bajo un árbol de raíz w, es la clasificación del nodo final de la arista que sale de w con la etiqueta que corresponde al valor de w para c.*

Notemos que esta definición muestra que el árbol con la operación de clasificar define una función recursiva y además que induce una partición sobre cada conjunto de críticas, por lo que satisface nuestra definición de Clasificador de críticas. Además cada árbol representa a una función del Espacio de hipótesis.

Uno de los algoritmos para generar árboles de decisión para un conjunto de ejemplares es el algoritmo C4.5 1 diseñado por el ingeniero australiano Ross Quinlan y que fue el sucesor del algoritmo para el mismo propósito ID3. Se trata de un algoritmo glotón [1], que etiqueta la raíz del árbol con el atributo que más información aporta (aquel de mayor entropía) y recursívamente construye los vecinos del nodo. Para una discusión más extensa de este proceso ver [**bell**][sección 18.3].

Justamente, para poder determinar sobre qué nodo del árbol conviene hacer la división se utiliza la *entropía de Shannon*, misma que mide la incertidumbre. Puede entonces también ser entendida como la medida de la *sorpresa* de una fuente de información; por consiguiente, mientras menos probable sea la ocurrencia de un evento determinado, mayor será su entropía asociada.

En general, dada una distribución de probabilidad $P = (p_1, p_2, \dots, p_n)$ y un conjunto de ejemplares S que toman los valores $x_1, x_2, \dots, x_n$ con probablidad p_i, $i \in \{1, \dots, n\}$, la entropía de la variable aleatoria dada por:[**russel**][pág. 704].

$$H(S) = -\sum_{i=1}^{n} p_i \times log_2(p_i)$$

La base del logaritmo puede variar. Dentro del terreno de las Ciencias de la Computación lo más común es usar base dos con tal de poder acumular *bits* de entropía: si un evento x_i es doblemente improbable que suceda, entonces la entropía se ve incrementada en log_2, que es equivalente a un *bit* de entropía de Shannon.

Luego de calculada la entropía, el algoritmo utiliza el concepto de *ganancia de información* (*information gain*) para decidir correctamente el nodo que representa el atributo sobre el que se hace la separación. La ganancia de información puede ser comprendida como el cambio existente entre la entropía de un estado previo a un nuevo estado en el que se tiene más información. A partir de la definición dada de entropía:

$$\Delta H(S) = H(S) - \frac{m_L}{m} H_L(S) - \frac{M_R}{m} H_R(S)$$

En donde m es el número total de instancias o ejemplares, m_k es el número de ejemplares cuyos valores pertenece a la clase k.

En particular, cuando se tienen tan sólo dos clases ($k = 2$ como en el caso del proyecto) y se tiene un clasificador binario, entonces la ganancia de información se basa en la entropía booleana del atributo objetivo del conjunto de ejemplares:

$$H(objetivo) = B(\frac{p}{p+n})$$

En donde $B(q)$ es la entropía booleana de una variable aleatoria que toma el valor de verdadero con una probabilidad q, así que aplicando la fórmula de entropía:

$$B(q) = -(qlog_2 q) + (1-q)log_2(1-q)$$

Luego, para determinar la ganancia de información de partir en un cierto nodo (atributo) A, tenemos que:

$$\Delta H(S) = Ganancia(A) = B(\frac{p}{p+n}) - Resto(A)$$

[1] En inglés *greedy*.

En donde $Resto(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k})$ representa la esperanza de información adquirida (entropía) de dividir el conjunto S de ejemplares en subconjunto $S_1, S_2, \ldots, S_d$ en donde cada subconjunto E_k tiene p_k instancias con un valor booleano positivo y n_k instancias con un valor booleano negativo.

La estrategia por supuesto se generaliza para $k > 2$ como se mencionó antes, teniendo que ir acumulando la entropía que resulta de considerar a cada atributo posible [**russel**][págs. 703-704].

Data: ejemplos¸ atributos, valor por defecto
Result: Árbol de decisión para el conjunto de ejemplares.
if *ejemplos es vacío* **then**
  regresar el valor por defecto;
end
if *todo los ejemplos tienen la misma clasificación* **then**
  regresar la clasificación;
end
if *No hay atributos* **then**
  regresar valor-mayoria(ejemplos);
end
b = mejor-atributo(atributos,ejemplos);
t = nuevo árbol de decisión con raíz a;
m = valor-mayoria(ejemplos);
for *valor* v_i *de a* **do**
  $ejemplos_i$ = elementos de ejemplos con $a = v_i$;
  subarbol = arbol-decision(ejemplos, atributos - b, m);
  agregar una rama a t etiquetada con v_i y árbol *subarbol*;
end

Algorithm 1: arbol-decision

Ventajas de los árboles de decisión

Los árboles de decisión son una de las formas más simples y eficientes de algoritmos de aprendizaje. La estructura de un árbol de decisión es fácil de interpretar, lo que permite hacer un análisis de la estructura de los datos. Esto les da otro punto de interés además de su utilidad como clasificadores.

Desventajas de los árboles de decisión

La construcción del algoritmo 1 puede resultar en Sobreajuste del árbol, una discusión de las técnicas que pueden ocuparse para intentar evitar este efecto se puede encontrar en [**quinlan**]. Una de las razones por las cuáles puede darse Sobreajuste es la expansión de atributos no-relevantes en el árbol. Si se da esta expansión el árbol tiende a memorizar las características del conjunto de entrenamiento y a perder la capacidad de generalizar. Para calcular la no-relevancia de un atributo se pueden usar tablas de χ^2 y podar los árboles usando esta información.

2.1.2. Redes Bayesianas

Definición 18 (Clasificador Bayesiano) *Un Clasificador Bayesiano (o Red Bayesiana) es una gráfica acíclica digida cuyos nodos son variables aleatorias y se cumplen algunas suposiciones de dependencia [***charniak***]. Los arcos representan la dependencia entre variables aleatorias.*

Especificar la distribución de probabilidad de una Red Bayesiana significa asignar probabilidades a todos los nodos raíz (aquellos sin predecesores) y las probabilidades condicionales de todos los nodos no-raíz considerando todas las combinaciones de sus predecesores. Los nodos no-raíz corresponden a las variables aleatorias correspondientes a los atributos y los nodos raíz la variable aleatoria correspondiente a la clase (es decir, la probabilidad de que un atributo tome un valor particular).

Una Red Bayesiana nos permite calcular las probabilidades condicionales de los nodos en la red, dado que se han observado los valores de algunos nodos. Esto nos da una descripción de las Redes Bayesianas como clasificadores.

Definición 19 (Evaluación de una Red Bayesiana) *Asignar los valores de los nodos que corresponden a los atributos nos permite calcular las probabilidades condicionales para los nodos que corresponden a clases. A esto se le llama evaluar la red.*

Al igual que con los árboles de decisión vemos que esto nos permite inferir la probabilidad de pertenencia de un ejemplar a una clase particular, y por lo tanto nos permite asignar a cada ejemplar una clase. Esto coincide con nuestra definición de Clasificador de críticas.

En este modelo las hipótesis son teorías probabilísticas de las características del dominio [**russell**][p. 802].

Frecuentemente se añade una suposición de independencia entre algunas variables, lo cuál motiva la siguiente definición.

Definición 20 (Clasificador Bayesiano ingenuo) *Un clasificador (o red) Bayesiana corresponde a un modelo de Bayes ingenuo cuando asume la independencia de todas las variables no-raíz. El tamaño de la representación crece linealmente sobre el número de nodos bajo esta suposición [***russell***][p. 809].*

Al igual que en la generación de los árboles de decisión, se puede dar el fénomeno de sobreajuste. Las Redes Bayesianas penalizan la complejidad de las hipótesis para mitigar este fenómeno [**russell**][p. 805].

Ventajas de las Redes Bayesianas

Las redes Bayesianas escalan bien para variables aleatorias binarias: cuando se tienen n atributos booleanos se tienen únicamente $2n + 1$ parámetros y no es necesario realizar una búsqueda para encontrar la hipótesis más probable [**russell**][p. 809]. Esto genera la expectativa de que tengan un buen desempeño en la clasificación de valores binarios.

En nuestro caso la hipótesis verdadera es un árbol de decisión y los clasificadores bayesianos no pueden representar con exactitud estas funciones, por lo que uno podría suponer que debería reflejarse un peor desempeño con Redes Bayesianas que con árboles de decisión. En los resultados veremos que no es una suposición apropiada y discutiremos el problema.

Los Clasificadores Bayesianos son óptimos cuando se computan las predicciones usando todas las hipótesis ponderadas con sus probabilidades, esto significa que dados valores para las variables aleatorias de los atributos cualquier otra predicción será correcta en menos ejemplares [**russell**][p. 804]. Esta optimalidad tiene un costo computacional y no se da en los clasificadores ingenuos en el caso general, esto los discutiremos en secciones posteriores.

Además las predicciones de las Redes Bayesianas se ajustan a la hipótesis verdadera *eventualmente* (al considerar su desempeño cuando el número de observaciones tiende a infinito).

Una ventaja de los clasificadores ingenuos de Bayes es que no tienen problema con tratar datos con ruido, o con información faltante [**russell**][p. 209]

Desventajas de las Redes Bayesianas

Las Redes Bayesianas dependen de contar y el número de cuentas que debe realizar incrementa mucho cuando se incrementan los nodos o variables de la red. Entre más nodos tenga la red más tareas debe realizar. Una red con 12 nodos, de los cuáles 7 tienen tres variables y los otros 5 tienen seis tendría que realizar 17,006,112 conteos, mientras que una con 3 nodos cada uno de 2 variables necesita sólo 8[**bell**].

En particular el cómputo exacto de probabilidades condicionales en una Red Bayesiana es un problema NP-DURO y durante un tiempo se suponía que aproximarlas era un problema polinomial, sin embargo se probó en 1993 que esta aproximación es otro problema NP-DURO, al igual que el cálculo exacto [**dagum**].

El uso de Redes Bayesianas en general se vuelve entonces restringido por estos resultados [**charniak**].

Sin embargo si se restringe a un Clasificador Bayesiano ingenuo la evaluación y el espacio empleado por la red es lineal sobre el número de nodos.

2.2. Descripción de la propuesta e implementación

Con tal de poder tener un número significativo de críticas para llevar a cabo el aprendizaje automatizado, decidimos en primera instancia obtener un conjunto de ejemplares (*dataset*) de el sitio Metacritic. Percatándonos que no todas las reseñas estaban completas en la página, pensamos en implementar un *web scraper* en lenguaje de programación `Python` para llevar a cabo el proceso de recolección de éstas. Sin embargo, como los críticos pueden publicar en diversas páginas asociadas a Metacritic, seleccionamos al crítico Mick LaSalle, quien nada más publica en SFGate.

La razón de la elección de el crítico se debió a su imparcialidad, no inclinándose a dar críticas positivas (o bien negativas) de las películas y sabiendo que tenía 2729 críticas.

Posteriormente, decidimos que sería conveniente tener más ejemplares para contemplar la opción de no centrarnos en un único crítico. Para esta parte obtuvimos los *datasets* del usuario albertbowden de `GitHub`.

Asimismo, recuperamos el `polarity dataset v2.0` ubicados en el apartado de experimentos para reconocimiento de sentimientos a través de críticas de películas de la Universidad de Cornell.

2.2.1. Obtención de ejemplares

Batch 1

Para poder obtener las críticas de Mick LaSalle se construyó un *web scraper* en `Python`. El archivo —mismo que se incluye como parte de los recursos de la implementación del proyecto— utiliza la biblioteca `BeautifulSoup` para poder analizar documentos `HTML`. De esta forma pudimos especificar las etiquetas que requeríamos para quedarnos con ellas fácilmente. Luego de tener los datos, los fuimos escribiendo paulatinamente en otro documento que especificamos en el código fuente.

En un segundo *script* `change_score.py` nos dedicamos exclusivamente a cambair las calificaciones de cada crítica de tal forma que no tuvieramos más que dos valores que representaran una crítica positiva (1) o una crítica negativa (0).

Batch 2

Luego de descargar el dataset, procedimos a abrilo con la aplicación `Weka`. Utilizamos el `TextDirectoryLoader` para poder cargar todos los archivos dentro del directorio (archivos con extensión de texto simple `.txt`), indicando que usara como nombres de clases los nombres de los subdirectorios. Luego de cargarlos en `Weka`, procedimos a gaurdar los datos ya con un formato `.arff` para poder usarlo posteriormente con `Java`.

Batch 3

El proceso de obtención fue análogo al del segundo grupo de instancias luego de haber clonado el repositorio de GitHub. Decidimos quedarnos solamente con el subconjunto de críticas del directorio `train` que estaban completamente etiquetadas para ser consistentes con los dos grupos de ejemplares previamente obtenidos en el sentido de que probaríamos los ejemplares con un *split* del archivo.

2.2.2. Conjunto de Datos

En los tres casos, los datos consisten de un atributo de tipo cadena (`String`) y otro numérico nominal que toma los valores cero o uno, acorde a la definición de crítica dada en un comienzo. Los archivos fueron llamados `Batch_1.arff`, `Batch_2.arff` y `Batch_3.arff`. Optamos por un uso del formato de archivos atributo-relacional para facilitar su manipulación con las bibliotecas de `Weka` desde nuestro proyecto de `Java` en `NetBeans`, además de que `Weka` nos facilitó en gran medida el proceso de conversión a este tipo de archivo.

El contenido de cada conjunto de ejemplares fue el siguiente:

Batch 1 2495 críticas de Mike Lasalle, con 1162 de ellas negativas y 1333 positivas.

Batch 2 2000 críticas de diversos críticos recopiladas por Bo Pang y Lillian Lee y utilizadas previamente en su artículo Thumbs up? Sentiment classification using machine learning techniques. La mitad de las críticas son negativas y la otra positivas.

Batch 3 25000 críticas positivas, con la mitad negativas y la otra mitad positivas en donde no se tienen más de treinta críticas de la misma película con tal de evitar valores de críticas que podrían estar correlacionados.

2.2.3. Trabajo en conjunto con `Java`

Para referise al proyecto, referirse a la carpeta de `Implementación` o bien al repositorio de GitHub.

Decidimos crear un proyecto con el enterno de desarrollo integrado `NetBeans` para poder enlazar las bibliotecas de `Weka` a través del `weka.jar` y entonces lograr una parametrización rápida de los dos algoritmos (*naivebayes* y *J48*), al igual que del filtro de los archivos con extensión `.arff`; en particular configuramos un filtro de tipo `StringToWordVector`.

El programa solicita al usuario el archivo con el que se realizará el entrenamiento y posterior prueba (*training* y *testing)* de los ejemplares. Es importante mencionar que estuvimos realizando algunas pruebas y notamos que la maner amás efectiva para que no haya problemas de rutas de los archivos es colocando el archivo `.arff` en el directorio del proyecto, pero no dentro de ninguna subcarpeta como `src`.

Luego de especificar el algoritmo con el cual llevar a cabo la clasificación, el usuario procede a dar algunos parámetros para configurar el filtro. En particular no dimos la alternativa de proporcionar la cadena de delimitadores

ni un *Stemmer* como el `Snowball` ya que suele haber problemas al enlazarlo con `Java`.

Luego de especificado todo lo anterior, el programa crea un filtro de tipo `StringToWordsVector` y se llama a un método que lo configura. Luego de calcular el tamaño que corresponde a entrenamiento y al de prueba a partir del porcentaje de *split* especificado por el usuario, se aplica a las instancias para obtener las dos instancias de prueba y entrenamiento, ya habiéndolas filtrado antes. En ambos casos se crea una evaluación con el conjunto de entrenamiento y se compara el desempeño con el conjunto de prueba, imprimiendo en cada caso el resumen adecuado.

Capítulo 3

Resultados

3.1. Resultados obtenidos

Los resultados que se mencionan a continuación corresponden a los obtenidos con nuestra implementación en `Java`, misma que fue congruente con los resultados de la aplicación `Weka` al usar correctamente sus bibliotecas.

Se agregan capturas de los resultados obtenidos en el archivo `resultados.pdf`[1]. Para poder replicar los experimentos con los argumentos, referirse al archivo `configuraciones.txt` del directorio `Documentación`.

3.1.1. Batch 1

J48

Notemos que el segundo caso fue el de peor desempeño del clasificador. Aquí no se usó la opción de igualar minúsuclas y mayúsculas, se usó el archivo de *stopwords*, no se usaron frecuencias, con mil palabras, un *confidence factor* de 0.25 y un 70 % de entrenamiento.

En contraparte, para el caso en que logró el máximo de 64 % de correcta clasificación la diferencia fue que sí se dio la opción de las minúsuculas, se usaron frecuencias y fue con 10000 palabras.

Naive Bayes

En el tercer caso se tuvo el peor caso con una clasificación correcta dle 70 % de las instancias, pero fue aún así mejor que el mejor caso del 64 % del uso del J48. Aquí usamos 1000 palabras, con un porcentaje de entrenamiento del 70 % y se usaron las frecuencias de las palabras.

Por otro lado, al no usar frecuencias y usando la misma configuración adicional, se logró el resultado del caso 6 con un éxito en el 77 % de la clasificación.

[1] Los ponemos aparte para poder referirse a la par que se leen los resultados.

3.1.2. Batch 2

J48

El peor caso en esta ocasión fue para el noveno caso de prueba, con un éxito del 55 %. Esto se debió claramente al entrenamiento del 30 % solamente. Dejando esta cuestión de lado, casi todos los casos fueron similares, con aproximadamente un 66 % de éxito sin importar demasiado la configuración del filtro. Logró bajar solamente un 1 % al no usar los stopwords y el peor caso dentro de los entrenamientos normales fue el de 63 % al haber usado 10000 palabras frente a las 1000 de los otro casos de prueba.

El mejor caso fue el quinto, con un 68 % de clasificación adecuada al usar todas las opciones de filtrado y 1000 palabras.

Naive Bayes

En el peor escenario presentado en el tercer caso de prueba, así como el cuarto, en donde se obtuvo 70 % de clasificación correcta; nuevamente fue mejor que el J48. En este caso también se usaron las frecuencias, siendo igual que en el peor caso del primer *batch*.

Por su parte, en prácticamente todos los demás casos se obtuivo una clasificación correcta de entre el 84 % y el 85 %; en ninguno de estos casos se usaron frecuencias.

3.1.3. Batch 3

J48

No hubo mucha diferencia entre los porcentajes de clasificación correcta en esta ocasión, oscilando todos entre el 73 % y 74 %.

Naive Bayes

Siguiendo el patrón encontrado previamente, el peor caso se dió el usar las frecuencias. En contraparte, al usar un filtro que no usara las frecuencias, sí usara todos los *tokens* en minúsculas y con un conjunto de entrenamiento del mismo tamaño que el previo (70 %), se incrementó el porcentaje de clasificación correcta cerca un 6 % entre el primer caso y el tercero.

3.1.4. Redes Bayesianas

También se probaron redes bayesianas como se muestra en `resultados.pdf`, pero no notamos una mejora significativa en relación con el clasificador bayesiano ingenuo.

3.1.5. Sobre el desempeño del J48

El clasificador basado en el C4.5 tuvo resultados de clasificación por debajo del clasificador bayesiano ingenuo. No obstante, el hecho de usar o no frecuencias no ocasionó que el algoritmo del J48 tuviera un peor desepeño en constraste con lo que sucedió con el clasificador bayesiano. Esta conclusión también fue a la que llegaron Sayed Mohsen, et al. en el año de 2013 en un proyecto de comparación de ambos algoritmos aplicados a *data mining* [**IJCSNS**].

El principal inconveniente que surge al utilizar árboles de decisión es el potencial efecto de que *memoricen* la información, logrando sobreajutarse (*overfitting*). También notamos que al no usar una poda, el sobreajuste es notable al usar conjuntos de entrenamiento de proporción pequeña frente a los de prueba.

Más aún, se sufre de la desventaja de que es necesario darle instancias de conjuntos de entrenamiento más grandes que los que requiere recibir un clasificador bayesiano para lograr una clasificación adecuada.

3.1.6. Sobre el desempeño del Clasificador Bayesiano ingenuo

Las hipótesis en las que trabaja el Clasificador Bayesiano ingenuo son particularmente fuertes, sin embargo su desempeño en la práctica muchas veces supera las expectativas. Esto ha fomentado investigación al respecto.

Zhang publicó condiciones suficientes para la optimalidad del clasificador ingenuo de Bayes [**zhang**]. Entre estas condiciones menciona que si las dependencias entre los atributos se distribuyen de forma uniforme en las clases o se cancelan entre sí, entonces el clasificador será óptimo sin importar el grado de dependencia entre los atributos. Un estudio de IBM en 2001 muestra experimentalmente que para distribuciones de baja entropía el clasificador ingenuo de Bayes ofrece resultados buenos [**ibm**].

Nuestros resultados sugieren que posiblemente las dependencias entre los atributos no se cancelan totalmente entre sí, ni que se distribuyen de forma uniforme, pero tal vez se den ambas condiciones de manera parcial.

La existencia de una palabra en el cuerpo de una crítica no ofrece mucha información y si se asignara una clase a la crítica basándose en este único atributo probablemente no podríamos decir mucho acerca de la probabilidad de errar. Esto invita a pensar que la distribución de las variables aleatorias no es de baja entropía en el problema.

Resulta interesante que la mejora con el Clasificador Bayesiano (no-ingenuo) no es significativa. Esto parece aportar evidencia a las hipótesis expuestas, pues posiblemente sea debido a que las dependencias entre los atributos se cancelan o están cerca de estar uniformemente distribuidas. Si fuera de otra forma el Clasificador Bayesiano (no-ingenuo) habría aprovechado que no hace las hipótesis de independencia y obtenido un desempeño significativamente mejor al del Clasificador Bayesiano ingenuo.

También destacamos que la inclusión del número de repeticiones en las palabras empeoró el desempeño del Clasificador Bayesiano ingenuo. Esto no necesariamente constituye evidencia de que dicha información no es relevante para el problema, podría ser el caso que incrementa la dependencia de las variables y por ello el desempeño del Clasificador Bayesiano ingenuo empeora.

3.2. Conclusiones

Se observó que la comprensión de las técnicas utilizadas, en particular la justificación de su funcionamiento, permite hacer una selección informada y efectiva de las herramientas y opciones que se ocuparán. También se apreció la importancia de tal comprensión en la interpretación de los resultados.

También se mostró que los resultados teóricos pueden tener utilidad en el momento de seleccionar técnicas, preprocesar los datos, seleccionar parámetros en los procesos, interpretar los resultados y ofrecer formas para mejorar los resultados.

Se observó que el campo del Reconocimiento de Patrones y Aprendizaje Automatizado está lo suficientemente maduro como para conectar los modelos matemáticos que se emplean para estudiarlo, así como sus resultados, con las aplicaciones prácticas que ofrece. Sin embargo el proceso aún es artesanal y depende mucho de probar, errar y cambiar la metodología, por lo que su automatización completa está limitada por este aspecto. Una pregunta que surge de este estudio es si tal limitación es inherente o se debe a que las técnicas no han madurado lo suficiente.

En particular se observó que el comportamiento del Clasificador Bayesiano ingenuo fue bueno de forma consistente, mientras que la construcción de árboles de decisión con el J48 en este caso no resultó muy eficiente.

El uso del aprendizaje automatizado es latente en la actualidad como pudimos apreciar en clases al discutir las aplicaciones tan diversas que se le ha dado a esta disciplina naciente. El poder calificar con base en una crítica escrita en lenguaje natural podría aportar información valiosa a la industria y evitar tentativamente una clasificación sesgada por parte de usuarios no experimentados al tener que dar ellos un valor nominal que logre resumir los sentimientos que pueden plasmar mejor en un texto.

Bibliografía

[1] Russell, S. J., Norvig, P., & Davis, E. (2016).
Artificial intelligence: a modern approach.
Harlow: Prentice Hall,
3rd Edition

[2] J. R. Quinlan,
Induction of Decision Trees,
Readings in Machine Learning,
Originally published in *Machine Learning*
1:81–106, 1986.

[3] Jason Bell,
Machine Learning, Hand-On for Professionals,
Wiley

[4] Moshkov, Zielosko,
Combinatorial Maching Learning, Springer

[5] Dagum, Paul and Luby, Michael. (1993).,
Approximating probabilistic inference in Bayesian belief networks is NP-hard.
Artificial Intelligence. 60. 141-153. 10.1016/0004-3702(93)90036-B.

[6] The Optimality of Naive Bayes,
Harry Zhang,
Faculty of Computer Science,
University of New Brunswic,
2004

[7] IBM Research Report,
An empirical study of the naive Bayes classifier,
Irina Rish,
2001

[8] Bayesian Networks without Tears,
Eugene Charniak

[9] A Dictionary of Statistics (3 ed.),
Graham Upton and Ian Cook,
Oxford University Press

[10] Evaluation of Naïve Bayes and J48 Learning Algorithm for Text Classification,
A. Sayed Mohsen Hashemi at al.
International journal of Computer Science & Network Solutions