

Sistemas de Recuperación de Información

Aplicaciones del
agrupamiento y
de la clasificación
en la recuperación
de información
en la Web

Autores

Laura Victoria Riera Pérez

Marcos Manuel Tirador del Riego

Resumen La recuperacin de informacin (RI) es un subcampo de la ciencia de la informacin relacionado con la representacin, el almacenamiento, acceso y recuperacin de la informacin. Las reas de investigacin actuales dentro del campo de RI incluyen bsquedas y consultas, clasificacin de resultados de bsqueda, navegacin y bsqueda de informacin, optimizacin de la representacin de informacin, almacenamiento, clasificacin de documentos y agrupamiento. A partir de la expansin y consolidacin de Internet, como medio principal de comunicacin electrnica de datos, se ha puesto a disposicin de casi toda la humanidad una importante cantidad de informacin de todo tipo. A los efectos de aprovechar todo este potencial de informacin, es necesario poseer accesos que permitan que la tarea de recuperacin sea eficiente y efectiva a la vez que se mejore la experiencia e interfaz de usuario. El objetivo principal de este artculo es comprender el uso de la agrupacin y clasificacin de documentos para mejorar la recuperacin de su informacin. De cada uno se muestran las características principales, medidas de similitud y evaluacin, ventajas, desventajas y las aplicaciones en la RI (en particular en la web). Además, se analizan los algoritmos K-Means y HAC de agrupación; y el Naive Bayes y K-Nearest-Neighbours de clasificación.

Palabras clave — agrupamiento · agrupamiento particionado · agrupamiento jerárquico · clasificación · web · Recuperación de Información (RI)

Índice general

1. Introducción	1
2. Agrupamiento	1
2.1. Medidas de similitud	2
2.2. Medidas de evaluación	2
2.3. Agrupamiento particionado	4
2.4. Agrupamiento jerárquico	6
2.5. Ventajas y desventajas	9
2.6. Ejemplos de aplicación	9
3. Clasificación	10
3.1. Naive Bayes	12
3.2. Feature Selection	15
3.3. K Nearest Neighbor	16
3.4. Medidas de evaluación	18
3.5. Ventajas y desventajas	19
3.6. Aplicaciones en la Recuperación de la Información.	20
3.7. Otros ejemplos de aplicación	20
4. Conclusiones	21

1. Introducción

La categorización es uno de los procesos fundamentales en la ciencia. Es importante reconocer que la categorización trasciende el intelecto humano y es una propiedad fundamental de los organismos vivos. Los intentos de desarrollar técnicas para separar observaciones de forma automática en grupos o clases requiere de la cuantificación de la similitud. Es por ello que la habilidad de reconocer dos objetos como más similares entre ellos que a un tercero de seguro debe haber estado en los ancestros de la raza humana. En muchos trabajos clasificatorios manuales es poco práctico estimar similitudes taxonómicas entre objetos de una muestra, debido entre otras cosas a las limitaciones al tiempo que le toma al ser humano realizar dicha tarea.

La clasificación y el agrupamiento son dos métodos de aprendizaje de máquinas usados para categorizar colecciones de datos. Ambos métodos tienen como objetivo separar las observaciones en grupos que compartan características similares, mientras se maximice la separación entre grupos que se diferencien mucho entre sí. Debido a que comparten un objetivo estos dos métodos suelen confundirse, sin embargo, se diferencian en la filosofía utilizada para la clasificación de los objetos. En clasificación los objetos se asignan a clases prefijadas de antemano de acuerdo a características perceptibles de los objetos. En cambio, en agrupamiento la separación en grupos ocurre de forma natural de acuerdo a similitudes entre los objetos que muchas veces el humano no es capaz de interpretar. Debido a la necesidad de un experto humano la clasificación entra dentro de la rama del aprendizaje supervisado, mientras que los algoritmos de agrupamiento son no supervisados.

2. Agrupamiento

Los algoritmos de agrupamiento, como su nombre lo indica, agrupan un conjunto de documentos en subconjuntos o clústeres. Son utilizados para generar una estructura de categorías que se ajuste a un conjunto de observaciones.

Los grupos formados deben tener un alto grado de asociación entre los documentos de un mismo grupo, es decir, deben ser lo más similares posibles, y un bajo grado entre miembros de diferentes grupos.

Es la forma más común de *aprendizaje no supervisado*, es decir no hay ningún experto humano que haya asignado documentos a clases. Es la distribución y composición de los datos lo que determina la pertenencia al clúster.

Si bien el agrupamiento a veces es denominado clasificación automática, esto no es estrictamente exacto ya que las clases formadas no se conocen antes de tratamiento, como implica la clasificación, sino que se definen por los elementos que se les asignan.

La entrada clave para un algoritmo de agrupamiento es la medida de distancia. Diferentes medidas de distancia dan lugar a diferentes agrupamientos. En el agrupamiento de documentos, la medida de distancia es a menudo también la distancia euclidiana.

Hipótesis de agrupamiento: *Los documentos en el mismo grupo se comportan de manera similar con respecto a la relevancia para las necesidades de informacin.*

La hiptesis, tomada de [[1], sección 16.1, p.350], establece que si hay un documento de un grupo que es relevante a una solicitud de bsqueda, entonces es probable que otros documentos del mismo clster tambien sean relevantes.

Los algoritmos de agrupamiento pueden ser clasificados según la pertenencia a los grupos en *agrupamiento exclusivo o fuerte* (hard clustering) donde cada documento es miembro de exactamente un grupo; o *agrupamiento difuso o suave* (soft clustering) donde un documento tiene membresa fraccionaria en varios grupos. También pueden clasificarse según el tipo de estructura impuesta sobre los datos como *agrupamiento particionado o plano* (flat clustering) o *agrupamiento jerárquico* (hierarchical clustering).

2.1. Medidas de similitud

Sean d_i el i -simo documento del corpus y w_{ik} el peso del término k de un total N ($N > 0$) en este documento.

■ Coeficiente de Dice:

$$S_{d_i, d_j} = \frac{2 \sum_{k=1}^N (w_{ik} w_{jk})}{\sum_{k=1}^N w_{ik}^2 + \sum_{k=1}^N w_{jk}^2}$$

■ Coeficiente de Jaccard:

$$S_{d_i, d_j} = \frac{\sum_{k=1}^N (w_{ik} w_{jk})}{\sum_{k=1}^N w_{ik}^2 + \sum_{k=1}^{max} w_{jk}^2 - \sum_{k=1}^N (w_{ik} w_{jk})}$$

■ Coeficiente del coseno:

$$S_{d_i, d_j} = \frac{\sum_{k=1}^N (w_{ik} w_{jk})}{\sqrt{\sum_{k=1}^N w_{ik}^2 \sum_{k=1}^N w_{jk}^2}}$$

Las ecuaciones anteriores fueron tomadas de [[3], sección 16.2.2, p.3].

2.2. Medidas de evaluación

- **Pureza:** Para calcular la pureza, cada grupo se asigna a la clase que es ms frecuente en el grupo, y luego se mide la precisin de esta asignacin contando el nmero de documentos correctamente asignados y dividiendo por N .

$$purity(\Omega, C) = \frac{1}{N} \sum_k max_j |\omega_k \cap c_j|$$

donde $\Omega = \{\omega_1, \omega_2, \dots, \omega_K\}$ es el conjunto de clústeres y $C = \{c_1, c_2, \dots, c_J\}$ es el conjunto de clases. Se interpreta ω_k como el conjunto de documentos en el clúster y c_j como el conjunto de documentos que pertenecen a esa clase. Esta ecuación fue tomada de [[1], ecuación 16.1, p.357].

Malos agrupamientos tienen valores de pureza cercanos a 0, mientras que un agrupamiento perfecto tiene una pureza 1.

La alta pureza es fcil de lograr cuando el nmero de grupos es grande, en particular, la pureza es 1 si cada documento tiene su propio grupo. Sin embargo en este caso se tendría la mayor cantidad de grupos posibles, por lo que la solución no tendría calidad.

- **Informacin mutua normalizada:** Una medida que permite hacer la compensación entre calidad y pureza es la informacin mutua normalizada o *NMI* por sus siglas en inglés.

$$NMI(\Omega, \mathbb{C}) = \frac{I(\Omega, \mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2}$$

Tomado de [[1], ecuación 16.2, p.358]

I es la información mutua:

$$I(\Omega, \mathbb{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)}$$

donde $P(\omega_k)$, $P(c_j)$ y $P(\omega_k \cap c_j)$ son las probabilidades de que un documento sea del clúster ω_k , clase c_j , y esté en la intersección de ω_k y c_j , respectivamente.

La ecuación anterior fue tomada de [[1], ecuación 16.3, p.358]

H es la entropía (mide la incertidumbre de una fuente de informacin)

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k)$$

La ecuación anterior fue tomada de [[1], ecuación 16.5, p.358]

Se alcanza la mxima informacin mutua para un agrupamiento exacto que perfectamente recrea las clases pero también si los grupos se subdividen en clústeres ms pequeños.

En particular, en un agrupamiento con $K = N$ los grupos de documentos tienen *MI* mximo. Entonces *MI* tiene el mismo problema que la pureza, no penaliza cardinalidades grandes.

La normalización por el denominador $[H(\Omega) + H(\mathbb{C})]/2$ soluciona este problema ya que la entropía tiende a aumentar con el nmero de clústeres, alcanzando su $\log N$ mximo para $K = N$, lo que asegura que *NMI* es bajo para una gran cantidad de clústeres.

- **Índice de Rand (Rand Index):** Se asignan dos documentos al mismo clster si y slo si son similares. Una decisin positiva verdadera (TP) asigna dos documentos similares al mismo grupo, una decisin negativa verdadera (TN) asigna dos documentos diferentes a diferentes grupos. Hay dos tipos

de errores que se pueden cometer. Una decisión de falso positivo (FP) asigna dos documentos no similares al mismo clster. Una decisión de falso negativo (FN) asigna dos documentos similares a diferentes agrupaciones. El índice de Rand (IR) mide el porcentaje de decisiones que son correctas (precisin).

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

La ecuación anterior fue tomada de [[1], sección 8.3, p.155].

El índice de Rand otorga el mismo peso a los falsos positivos y falsos negativos. Separar documentos similares a veces es peor que poner pares de dos documentos no similares en el mismo grupo.

- **Medida F:** Se puede usar la medida F, vista anteriormente en conferencia, para penalizar los falsos negativos ms fuertemente que los falsos positivos seleccionando un valor $\beta > 1$, dando as ms peso al recobrado.

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

La ecuación anterior fue tomada de [[1], sección 8.3, p.154].

2.3. Agrupamiento particionado

El *agrupamiento particionado* crea un conjunto de clsteres sin ninguna estructura explícita que los relacione entre s.

Algoritmo K-means

Es el algoritmo de agrupamiento plano ms importante. Su objetivo es minimizar la distancia euclidiana al cuadrado promedio entre los documentos y el centro de sus clústeres.

El centro de un clúster se define como la media o centroide μ de los documentos en un grupo ω :

$$\vec{\mu}(\omega) \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$$

La ecuación anterior fue tomada de [[1], capítulo 6, p.131].

Se asume que los documentos se representan como vectores de longitud normalizada en un espacio de valor real de la manera habitual.

Una medida de qu tan bien los centroides representan a los miembros de su clúster es la suma residual de cuadrados o RSS, que es la distancia al cuadrado de cada vector desde su centroide sumado sobre todos los vectores:

$$RSS_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

$$RSS = \sum_{k=1}^K RSS_k$$

La ecuación anterior fue tomada de [[1], ecuación 16.7, p.360].

RSS es entonces la funcin objetivo en K-means y nuestro objetivo es minimizarla.

Algoritmo 1 K-Means

Parámetros de entrada: $\{\vec{x}_1, \dots, \vec{x}_N\}, K$

```

1:  $(\vec{s}_1, \dots, \vec{s}_K) \leftarrow \text{SelectRandomSeeds}(\{\vec{x}_1, \dots, \vec{x}_N\}, K)$ 
2: for  $k \leftarrow 1$  to  $K$  do
3:    $\vec{\mu}_k \leftarrow \vec{s}_k$ 
4: while no se cumpla la condición de parada do
5:   for  $k \leftarrow 1$  to  $K$  do
6:      $\omega_k \leftarrow \{\}$ 
7:     for  $n \leftarrow 1$  to  $N$  do
8:        $j \leftarrow \arg \min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$ 
9:        $\mu_j \leftarrow \omega_j \cup \{\vec{x}_n\}$  (reassignando los vectores)
10:    for  $k \leftarrow 1$  to  $K$  do
11:       $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$  (reeligiendo las semillas)
12: return  $\{\vec{\mu}_1, \dots, \vec{\mu}_N\}$ 

```

En esta implementación de K-means, tomada de [[1], figura 16.5, p.361], el primer paso es seleccionar al azar como centros iniciales de los clústeres a K documentos, estas son las semillas. Luego, el algoritmo mueve los centros de los grupos en el espacio para minimizar el RSS. Este proceso se repite de manera iterativa hasta que se cumpla un criterio de parada.

Opciones para la elección de las semillas:

- Escoger al azar.
- Calcular la media m de todos los datos y generar k centros iniciales agregando un pequeño vector aleatorio a la media ($m \pm \varsigma$).
- Utilizar otro mtodo como un algoritmo de agrupamiento jerárquico en un subconjunto de los objetos.

Algunos criterios de parada:

- Cuando se ha completado un número fijo de iteraciones I. Esta condición limita el tiempo de ejecución del algoritmo de agrupamiento, pero en algunos casos la calidad de el agrupamiento ser deficiente debido a un número insuficiente de iteraciones.
- Cuando la asignación de documentos a grupos no cambia entre iteraciones. Excepto en los casos con un mínimo local malo, esto produce un buen agrupamiento, pero los tiempos de ejecución pueden ser demasiado largos.

- Cuando los centroides no cambian entre iteraciones.
- Cuando RSS cae por debajo de un umbral. Este criterio asegura que el agrupamiento tiene la calidad deseada después de la terminación. En la práctica, es necesario tener además un límite en el número de iteraciones para garantizar terminación.

2.4. Agrupamiento jerárquico

El *agrupamiento jerárquico* produce una jerarquía, una estructura que es más informativa que el conjunto no estructurado de clústeres devuelto por el agrupamiento particionado, no requiere que se especifique previamente el número de grupos y la mayoría son deterministas, sin embargo son más ineficientes que los particionados.

Los algoritmos de agrupamiento jerárquico pueden tener dos enfoques: de abajo hacia arriba (bottom-up) llamados de *agrupamiento jerárquico aglomerativo* o de arriba hacia abajo (top-down) conocidos como de *agrupamiento jerárquico divisivo*.

Agrupamiento jerárquico aglomerativo

Los algoritmos de *abajo hacia arriba* tratan cada documento como un clúster nuevo desde el principio y luego fusionan (o aglomeran) sucesivamente pares de grupos hasta que todos los grupos se han fusionado en uno solo que contiene todos los documentos. Es por esto que se denomina agrupamiento jerárquico aglomerativo o HAC por sus siglas en inglés.

Toman decisiones basadas en patrones locales sin tener inicialmente en cuenta la distribución global. Estas decisiones tempranas no se pueden deshacer.

Medidas de similitud para clústeres en HAC

- **Agrupamiento por enlazamiento único** (single link clustering): La similitud entre dos clústeres es la similitud de los dos objetos más cercanos entre ellos (mayor similitud).

$$\text{sim}(\omega_i, \omega_j) = \max_{\vec{x} \in \omega_i, \vec{y} \in \omega_j} \text{SIM}(\vec{x}, \vec{y})$$

La ecuación anterior fue tomada de [[2], p.50].

- **Agrupamiento por enlazamiento completo** (complete link clustering): La similitud entre dos clústeres es la similitud de los dos objetos más alejados entre ellos (menor similitud).

$$\text{sim}(\omega_i, \omega_j) = \min_{\vec{x} \in \omega_i, \vec{y} \in \omega_j} \text{SIM}(\vec{x}, \vec{y})$$

La ecuación anterior fue tomada de [[2], p.51].

- **Agrupamiento aglomerativo por promedio de grupo** (group-average agglomerative clustering): El agrupamiento aglomerativo por promedio de grupo o GAAC por sus siglas en inglés, calcula la similitud promedio SIM-GA de todos los pares de documentos, incluidos los pares del mismo grupo (las auto-similitudes no están incluidas en el promedio).

$$SIM - GA(\omega_i, \omega_j) = \frac{1}{(N_i + N_j)(N_i + N_j - 1)} \sum_{\vec{x} \in \omega_i \cup \omega_j} \sum_{\vec{y} \in \omega_i \cup \omega_j, \vec{x} \neq \vec{y}} \vec{x} \cdot \vec{y}$$

La ecuación anterior fue tomada de [[1], ecuación 17.1, p.389].

Esta evalúa la calidad de un clúster basada en todas las similitudes entre documentos, evitando así castigar valores extremos como en los criterios de enlace nico y enlace completo, que establecen la similitud del clúster con la similitud de un solo par de documentos.

- **Agrupamiento por centroide** (centroid clustering): La similitud de dos clústers está definida como la similitud de sus centroides

$$\begin{aligned} sim(\omega_i, \omega_j) &= \mu(\omega_i) \cdot \mu(\omega_j) \\ &= \left(\frac{1}{N_i} \sum_{\vec{x} \in \omega_i} \vec{x} \right) \cdot \left(\frac{1}{N_j} \sum_{\vec{y} \in \omega_j} \vec{y} \right) \\ &= \frac{1}{N_i N_j} \sum_{\vec{x} \in \omega_i} \sum_{\vec{y} \in \omega_j} \vec{x} \cdot \vec{y} \end{aligned}$$

La ecuación anterior fue tomada de [[1], ecuación 17.6, p.391].

A diferencia de GAAC, el agrupamiento por centroide excluye el cálculo de pares del mismo clúster.

Algoritmo HAC

Dado un conjunto de N elementos a agrupar, el proceso básico del agrupamiento jerárquico es:

1. Se comienza con N clústeres, resultado de asignar cada elemento al suyo propio. Se computa la matriz C de similitud de NN.
2. Se halla la similitud entre los pares de clústeres con la medida deseada.
3. Se toma el par más similar de clústeres y se combinan en un nuevo clúster.
4. Se calculan las similitudes entre el nuevo clúster y cada uno de los clústeres antiguos.
5. Se repiten los pasos 3 y 4 hasta que todos los elementos estén agrupados en un solo grupo de tamaño N.

Algoritmo 2 HACParámetros de entrada: $\{d_1, \dots, d_N\}$

```

1: for  $n \leftarrow 1$  to  $N$  do
2:   for  $i \leftarrow 1$  to  $N$  do
3:      $C[n][i] \leftarrow SIM(d_n, d_i)$ 
4:      $I[n] \leftarrow 1$  (lleva los clústeres activos)
5:    $A \leftarrow []$  (secuencia de mezclas aplicadas en el agrupamiento)
6: for  $k \leftarrow 1$  to  $N - 1$  do
7:    $\langle i, m \rangle \leftarrow \arg \max_{\langle i, m \rangle: i \neq m \wedge I[i]=1 \wedge I[m]=1} C[i][m]$ 
8:    $A.APPEND(\langle i, m \rangle)$  (almacena la mezcla)
9:   for  $j \leftarrow 1$  to  $N$  do
10:     $C[i][j] \leftarrow SIM(i, m, j)$ 
11:     $C[j][i] \leftarrow SIM(i, m, j)$ 
12:     $I[m] \leftarrow 0$  (desactiva el clúster)
13: return  $A$ 

```

Esta implementación del algoritmo HAC fue tomada de [[1], figura 17.2, p.381].

Notas:

- En cada iteración, los dos clústeres más similares se fusionan y las filas y columnas del clúster fusionado i en C se actualizan.
- El agrupamiento se almacena como una lista de fusiones en A .
- I indica qué clústeres están disponibles para fusionarse.
- La función $SIM(i, m, j)$ calcula la similitud del grupo j con la fusión de los grupos i y m .

Una suposición fundamental en HAC es que la operación de fusión es monótona, es decir si s_1, s_2, \dots, s_{K-1} son las similitudes de combinación de las fusiones sucesivas de un HAC, entonces se cumple $s_1 \geq s_2 \geq \dots \geq s_{K-1}$.

El agrupamiento jerárquico no requiere un número predeterminado de clústeres. Sin embargo, en algunas aplicaciones se quiere una partición de clústeres disjuntos como en el agrupamiento particionado. En esos casos, la jerarquía debe cortarse en algún momento. Se pueden utilizar varios criterios para determinar el punto de corte:

- Cortar a un nivel de similitud preespecificado.
- Al igual que en el agrupamiento particionado, también se puede preespecificar el número de agrupamientos K y seleccionar el punto de corte que produce K clústeres.

Agrupamiento jerárquico divisivo

Los algoritmos de *arriba hacia abajo* comienzan con todos los documentos en un grupo. El clúster se divide utilizando un algoritmo de agrupamiento particionado. Este procedimiento se aplica recursivamente hasta que cada documento esté en su propio clúster.

A pesar de necesitar un segundo algoritmo de agrupamiento particionado como una subrutina, tiene la ventaja de ser ms eficiente si no se genera una jerarquía completa hasta las hojas de documentos individuales. Además, se beneficia de la informacin completa sobre la distribucin global al tomar decisiones de particin de alto nivel.

2.5. Ventajas y desventajas

El agrupamiento juega un papel fundamental en la recuperacin de informacin, sin embargo, no es siempre la respuesta a los problemas de este tipo. Debemos tener en cuenta sus ventajas y desventajas para utilizarlo correctamente, en el momento apropiado.

Ventajas:

- No es necesario identificar las clases antes del procesamiento por lo que no se debe contar con expertos para este fin.
- Es til para proporcionar estructura en grandes conjuntos de datos multivariados.
- Se ha descrito como una herramienta de descubrimiento porque tiene el potencial para revelar relaciones previamente no detectadas basadas en datos complejos.
- Debido a su amplia aplicacin en disímiles campos, cuenta el apoyo de una serie de paquetes de software, a menudo disponibles en la informática académica y otros entornos, por lo que se facilita su utilizacin.

Desventajas:

- No se tiene una idea exacta de las clases creadas.
- No recibe retroalimentacin.

2.6. Ejemplos de aplicacin

El agrupamiento es una técnica importante para descubrir subregiones o subespacios relativamente densos de una distribucin de datos multidimensional. Se ha utilizado en la recuperacin de informacin para muchos propsitos diferentes, como la expansin de consultas, la agrupacin e indexacin de documentos y la visualizacin de resultados de bsqueda. Permiten mejorar interfaz y experiencia de usuario y proporcionar una mayor eficacia o eficiencia del sistema de bsqueda.

A continuacin se describen con más detalle algunas de las aplicaciones más importantes en la web:

- Agrupamiento de resultados de bsqueda (Search result clustering): La presentacin predeterminada de los resultados de bsqueda (documentos devueltos en respuesta a una consulta) en la recuperacin de informacin es una lista sencilla. Los usuarios escanean la lista de arriba a abajo hasta que encuentran la informacin que buscan.

En su lugar, en la agrupación en clústeres de resultados de búsqueda los documentos similares aparecen juntos, siendo más fácil escanear algunos grupos coherentes que muchos documentos individuales. Esto es particularmente útil si un término de búsqueda tiene diferentes significados.

- **Dispersión-recopilación (Scatter-Gather):** Su objetivo es también una mejor interfaz de usuario. Este agrupa toda la colección para obtener grupos de documentos que el usuario puede seleccionar o reunir manualmente. Los grupos seleccionados se fusionan y el conjunto resultante se vuelve a agrupar. Este proceso se repite hasta que se encuentre un grupo de interés.

La navegación basada en la agrupación de clústeres es una alternativa interesante a la búsqueda por palabras clave, el paradigma de recuperación de información estándar. Esto es especialmente cierto en escenarios donde los usuarios prefieren navegar en lugar de buscar porque no están seguros de qué términos utilizar en la búsqueda.

- **Modelado de lenguaje (Language modeling):** Explora directamente la hipótesis del agrupamiento para mejorar los resultados de búsqueda, basado en una agrupación de toda la colección. Se usa un índice invertido estándar para identificar un conjunto inicial de documentos que coincide con la consulta, pero luego se agregan otros documentos de los mismos grupos, incluso si tienen poca similitud con la consulta.

Para evitar problemas de datos escasos en el modelado de lenguaje enfocado a la RI, el modelo de documento d se puede interpolar con un modelo de colección. Pero la colección contiene muchos documentos con términos atípicos de d . Al reemplazar el modelo de colección con un modelo derivado del grupo de d , se obtienen estimaciones más precisas de las probabilidades de ocurrencia de términos en d .

- **Recuperación basada en clústeres (Cluster-based retrieval):** La agrupación también puede acelerar la búsqueda.

La búsqueda en el modelo de espacio vectorial equivale a encontrar los vecinos más cercanos a la consulta. El índice invertido admite la búsqueda rápida del vecino más cercano para la configuración estándar de la RI. Sin embargo, a veces es posible que no se pueda usar un índice invertido de manera eficiente. En tales casos, se podría calcular la similitud de la consulta con cada documento, pero esto es lento. La hipótesis de agrupamiento ofrece una alternativa: encontrar los clústeres que están más cerca de la consulta y solo considerar los documentos de estos. Como hay muchos menos clústeres que documentos, se disminuye grandemente el espacio de búsqueda, y encontrar el clúster más cercano es rápido. Además, los elementos que coinciden con una consulta son similares entre sí, por lo que tienden a estar en los mismos clústeres, de esta forma la calidad no disminuye en gran medida.

3. Clasificación

El problema de clasificación en sentido general consiste en determinar dentro de un conjunto de clases a cuál de ellas pertenece un objeto dado. En el marco de este documento es de interés estudiar la clasificación de textos.

La forma más antigua de llevar a cabo la clasificación es manualmente. Por ejemplo, los bibliotecarios clasifican los libros de acuerdo a ciertos criterios, de modo que encontrar una información buscada no resulte una tarea de gigante dificultad. Sin embargo, la clasificación manual tiene sus límites de escalabilidad.

Como alternativa podría pensarse el uso de *reglas* para determinar automáticamente si un texto pertenece o no a una clase determinada de documentos.

Se ilustrará un ejemplo de regla aplicada automáticamente. Supongamos que un usuario necesita hacer una consulta en una página de noticias, por ejemplo, necesita tener actualidad sobre noticias relacionadas con las finanzas para tomar decisiones en su negocio. Al usuario entonces le podría interesar que de hacer una consulta en el sistema dicha consulta se mantenga ejecutando y le provea periódicamente las noticias relativas a finanzas.

A este tipo de consultas se le denomina consultas permanentes (o *standing queries* en inglés). Una consulta permanente es aquella que se ejecuta periódicamente en una colección a la cual nuevos documentos se adicionan en el tiempo. Toda consulta permanente se puede ver como un tipo de regla que se aplica a un sistema de clasificación que divide una colección en documentos que satisfacen la consulta y documentos que no.

Una regla captura una cierta combinación de palabras claves que identifican una clase. Reglas codificadas a mano pueden llegar a ser altamente escalable, pero crearlas y mantenerlas requiere un elevado costo en recursos humanos.

Existe, sin embargo, un enfoque adicional a los dos anteriores mencionados. Este es el uso de *Aprendizaje de Máquinas*. En este enfoque el conjunto de reglas de clasificación, o en general, el criterio usado para clasificar, es aprendido de forma automática a partir de los datos de entrenamiento.

Al uso del Aprendizaje de Máquinas en la clasificación de textos se le conoce como clasificación estadística de texto (o en inglés *statistical text classification*) si el método de aprendizaje usado es estadístico.

Se presenta a continuación la definición formal del problema de clasificación de textos, en el contexto del Aprendizaje de Máquinas.

Definición 1 Sea \mathcal{X} el espacio de documentos y $\mathcal{C} := \{c_i \mid c_i \subset \mathcal{X}, i \in \{1, 2, \dots, n\}\}$ un conjunto fijo de clases (también llamadas categorías o etiquetas). Sea además D un conjunto entrenado de documentos clasificados $(d, c) \in \mathcal{X} \times \mathcal{C}$. El problema de la clasificación de textos consiste en encontrar, usando métodos o algoritmos de aprendizaje, una función clasificadora $\gamma : \mathcal{X} \rightarrow \mathcal{C}$, que mapee documentos a clases, que satisfaga que $D \subset \gamma$.

El aprendizaje que toma parte en la búsqueda de γ es llamado *aprendizaje supervisado* debido a que se necesita la ayuda de uno o varios expertos que creen el conjunto de entrenamiento D . Estos expertos son también quienes determinan el conjunto de clases en que se clasificarán los textos. Se denotará por Γ el método de aprendizaje supervisado descrito, el cual actúa como una función que mapea un conjunto de datos de entrenamiento en una función clasificadora, o sea que $\Gamma(D) = \gamma$.

La definición 1 implica que cada documento pertenece a una sola clase. Pero existe otro tipo de problemas que permiten que un documento pertenezca a más de una clase. Por ahora este trabajo se enfocará en el tipo de una clase.

3.1. Naive Bayes

Uno de los métodos más comunes de aprendizaje supervisado es el conocido como *Naive Bayes* (NB). Este es un método de aprendizaje probabilístico. La probabilidad de un documento d de pertenecer a una clase c se puede expresar como $P(c | d)$. La tarea del algoritmo es encontrar la mejor clase para cada documento d . Para ello NB establece que la clase más apropiada para un documento es la más probable, o sea

$$c_{map} = \arg \max_{c \in \mathcal{C}} P(c | d).$$

La clase escogida para d se denota por c_{map} debido a que este método de clasificación, de acuerdo a la clase más probable para un documento dado, es conocido como *maximum a posteriori* (MAP).

Sin embargo la probabilidad condicional $P(c | d)$ es difícil de determinar. Haciendo uso del *Teorema de Bayes* la probabilidad anterior puede ser expresada como

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}.$$

El factor de normalización $P(d)$ es usualmente ignorado ya que no aporta información a la hora de buscar la clase más apropiada para un documento d , ya que este tiene el mismo efecto en todos los candidatos. Este cálculo puede ser simplificado si lo expresamos en términos de los términos en los documentos. Supongamos que $\{t_1, t_2, \dots, t_n\}$ son los términos que aparecen en d . Entonces tenemos que

$$c_{map} = \arg \max_{c \in \mathcal{C}} P(c)P(d | c) = \arg \max_{c \in \mathcal{C}} P(c) \prod_{1 \leq k \leq n} P(t_k | c),$$

donde $P(t_k | c)$ es la probabilidad de que el término t_k aparezca en un documento de la clase c . Podemos considerar $p(t_k | c)$ como una medida de qué tanto demuestra el término t_k que c es la clase correcta. El término $P(c)$ es conocido como probabilidad previa (*prior probability*) y en caso de que la información aportada por los términos no sea determinante en la selección podemos siempre escoger la clase con mayor valor de $P(c)$.

Para simplificar aun más el cómputo se pueden sustituir los valores anteriores por sus logaritmos. Esto reducirá el costo de hacer los cálculos y además los errores aritméticos, dado que la multiplicación se transforma en suma. La clase seleccionada sería entonces

$$c_{map} = \arg \max_{c \in \mathcal{C}} \left(\log(P(c)) + \sum_{1 \leq k \leq n} \log(P(t_k | c)) \right).$$

Solo queda ver como se estiman los parámetros $P(c)$ y $P(t_k | c)$, dado que los valores reales no son posibles de calcular. Para la probabilidad previa se puede contar la frecuencia relativa de cada clase en D :

$$P(c) = \frac{N_c}{N},$$

donde N_c es el número de documentos en la clase c y N es el número total de documentos. Se procede de manera similar para la probabilidad específica de una palabra en una clase

$$P(t_k | c) = \frac{T_{c,t_k}}{T_c},$$

donde T_{c,t_k} indica la cantidad de veces que ocurre la palabra t_k en todos los documentos de la clase c y T_c es la cantidad total de palabras (contando repeticiones) en toda la clase c . Si embargo, aún tenemos un problema con estas fórmulas y es que estamos asignando probabilidad cero a todos las clases que no contengan a todas las palabras del documento a clasificar. Para evitar esto adicionamos por defecto una unidad a cada contador lo cual es conocido como *Laplace smoothing*

$$P(t | c) = \frac{T_{c,t} + 1}{T_c + |V|},$$

donde $|V|$ es el número total de términos en el vocabulario.

Es importante destacar que en este método se está obviando la posición de las palabras. Se presentan aquí los Algoritmos 3 y 4 para entrenar y clasificar usando BN que fueron tomados de [1, Figure 13.2].

Algoritmo 3 TrainMultinomial

Parámetros de entrada: Conjunto de clases \mathcal{C} y conjunto de entrenamiento D .

- 1: $V \leftarrow \text{ExtractVocabulary}(D)$
- 2: $N \leftarrow \text{CountDocs}(D)$
- 3: **for** $c \in \mathcal{C}$ **do**
- 4: $N_c \leftarrow \text{CountDocsInClass}(D, c)$
- 5: $\text{prior}[c] \leftarrow N_c/N$
- 6: $\text{text}_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(D, c)$
- 7: **for** $t \in V$ **do**
- 8: $T_{ct} \leftarrow \text{CountTokensOfTerm}(\text{text}_c, t)$
- 9: **for** $t \in V$ **do**
- 10: $\text{condprob}[t][c] \leftarrow \frac{T_{c,t}+1}{\sum_{t'} (T_{c,t'}+1)}$
- 11: **return** $V, \text{prior}, \text{condprob}$

Algoritmo para entrenar Naive Bayes tomado de [1, Figura 13.2]

Algoritmo 4 ApplyMultinomialNB

Parámetros de entrada: $\mathcal{C}, V, prior, condprob, d$

```
1:  $W \leftarrow ExtractTokensFromDoc(V, d)$ 
2: for  $c \in \mathcal{C}$  do
3:    $score[c] \leftarrow \log prior[c]$ 
4:   for  $t \in W$  do
5:      $score[c] += \log condprob[t][c]$ 
6: return  $\arg \max_{c \in \mathcal{C}} (score[c])$ 
```

Algoritmo para aplicar Naive Bayes tomado de [1, Figura 13.2]

Podemos deducir de los algoritmos que la complejidad de ambos es lineal en el tiempo que toma escanear la información. Dado que esto hay que hacerlo al menos una vez, se puede decir que este método tiene complejidad temporal óptima. Dicha eficiencia hace que NB sea un método de clasificación tan usado.

3.2. Feature Selection

Un término con ruido (*noise feature*) es aquel que al pertenecer a la representación de los documentos, provoca un aumento del error de clasificación de los datos. Por ejemplo, supóngase que se tiene una palabra que ocurre rara vez, pero que en el conjunto de entrenamiento ocurre siempre en la misma clase c . Entonces al clasificar un documento nuevo que contiene esta palabra, la misma provocará que el clasificador se incline en cierta medida por seleccionar esta a c como respuesta. Sin embargo, dado que la ocurrencia de esta palabra únicamente en c es accidental, claramente no aporta información suficiente para la clasificación y por tanto, al considerar lo contrario, aumenta el error.

Este es uno de los propósitos que tiene la selección de términos (*feature selection* (FS)). Esta consiste en reducir el vocabulario, considerado en la clasificación de textos solo un subconjunto del que aparece en el conjunto de entrenamiento. Nótese que, al disminuir el tamaño del vocabulario aumenta la eficiencia de los métodos de entrenamiento y clasificación (aunque no es el caso de NB).

Selección de términos prefiere un clasificador más simple antes que uno más complejo. Esto es útil cuando el conjunto de entrenamiento no es muy grande.

En FS usualmente se fija una cantidad k de vocablos por cada clase c , que serán los usados por el clasificador. Para seleccionar los k términos deseados se establece un ranking entre los términos de la clase, haciendo uso de una función de medida de utilidad $A(t, c)$, y luego se escogen los k mejor posicionados. El algoritmo básico consiste en para cada clase c iterar por todos los términos del vocabulario y computar su medida de utilidad para la clase; para finalmente ordenar los resultados y devolver una lista con los k mejores.

Se presenta a continuación tres de los métodos de calcular $A(t, c)$ más comunes.

■ Información Manual.

Computar $A(t, c)$ como el valor esperado de información mutua (*Mutual Information* (MI)), da una medida de cuánta información aporta, la presencia en c de un término dado, a tomar la decisión correcta de clasificación de un documento. La siguiente definición fue tomada de [1, Ecuación 13.16]

$$I(U_t; C_t) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U_t = e_t, C_t = e_c) \log_2 \frac{P(U_t = e_t, C_t = e_c)}{P(U_t = e_t)P(C_t = e_c)},$$

donde U_t es una variable aleatoria que toma valor $e_t = 1$ si el documento contiene el término t y $e_t = 0$ en otro caso, y C es otra variable aleatoria que toma valor $e_c = 1$ si el documento está en la clase c y $e_c = 0$ en otro caso.

MI mide cuánta información un término contiene acerca de una clase. Por tanto, mantener los términos que están cargados de información, y eliminar los que no, contribuye a reducir el ruido y mejorar la precisión del clasificador.

■ Selección Chi cuadrado χ^2 .

En estadística se dice que dos eventos son independientes si el resultado de uno no afecta al resultado del otro. Esto se puede escribir formalmente

como $P(AB) = P(A)P(B)$. En estadística el test χ^2 se usa para medir el grado de independencia de dos eventos. En FS se puede entonces considerar aplicar este test asumiendo como eventos la ocurrencia de los términos y la ocurrencia de las clases. La siguiente definición fue tomada de [1, Ecuación 13.18]

$$\chi^2(D, t, c) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}},$$

donde N es la frecuencia según D , E es la frecuencia esperada y e_t y e_c se definen como en la medida anterior.

■ **Selección basada en frecuencia.**

Esta medida consiste en priorizar los términos que son más comunes en la clase. Puede ser calculada de dos formas diferentes. La primera es cantidad de repeticiones de un término en los documentos de una clase, conocida como frecuencia en colección. La otra es frecuencia de documentos, y se calcula como la cantidad de documentos en la clase que contienen al término en cuestión.

Cuando son seleccionados varios miles de términos, entonces esta medida es bastante buena. Esta es preferible a otros métodos más complejos cuando se aceptan soluciones subóptimas.

3.3. K Nearest Neighbor

En el algoritmo de Naive Bayes se representaron los documentos como vectores booleanos de términos. Luego se analizó que hay términos que no eran relevantes y que aportaban ruido, cosa que se solucionó seleccionando para el clasificador solamente un subconjunto de todos los términos. Aún así estamos clasificando la relevancia de cada término de manera binaria en relevante o no relevante (que aporta ruido).

El método que será presentado en esta sección, así como otros similares, asignan a cada término cierto valor de importancia relativa al documento en que aparece. Para esto se cambia la representación de los documentos a vectores de $\mathbb{R}^{|V|}$, donde a cada componente corresponde cierto peso que se le asigna al término correspondiente a esta. Entonces, el espacio de documentos \mathcal{X} (dominio de γ) es $\mathbb{R}^{|V|}$. A esta forma de representación de documentos se le conoce como modelo de espacio de vectores. La hipótesis básica para usar el modelo de espacio de vectores se presentará a continuación y fue tomada de [1, p. 289].

Hipótesis de contigüidad: Documentos en la misma clase forman una región contigua y regiones de diferentes clases no se superponen.

Las decisiones de muchos clasificadores basados en espacio de vectores dependen de una noción de distancia. Pueden ser usadas por ejemplo similitud basado en el coseno (del ángulo formado entre los vectores) o distancia Euclídeana. Por lo general, no hay mucha diferencia entre usar una u otra de estas distancias.

La tarea de la clasificación en el modelo de espacio de vectores es determinar las fronteras entre los documentos pertenecientes a una u otra clase. Estas últimas son llamadas fronteras de decisión ya que dividen el espacio en diferentes

poliedros, tales que si un documento pertenece a uno determinado, automáticamente sabemos de qué clase es.

En K Nearest Neighbor la frontera de decisión se determina localmente. En este se asigna cada documento a la misma clase que la mayoría de los k puntos más cercanos al documento. Basado en la hipótesis de contigüidad se espera que el documento d pertenezca a la misma clase que aquellos más cercanos a él.

En kNN para subdividir el espacio de documentos en regiones, dado un $k \in \mathbb{N}$ fijo, consideramos cada región como el conjunto de puntos para los cuales los k puntos más cercanos son los mismos. Estas regiones son poliedros convexos. Luego para cada una de estas regiones existe una clase a la que pertenecen todos sus puntos, que es aquella a la que pertenecen la mayoría de los documentos, ya clasificados, que están dentro de la región. En caso de que haya empate, la decisión de a qué clase asignar a un nuevo documento que pertenece a esta región del espacio, es tomada aleatoriamente entre las clases empatadas. Los algoritmos 5 y 6 corresponden a los métodos de entrenamiento y aplicación de kNN .

Algoritmo 5 Train- kNN

Parámetros de entrada: Conjunto de clases \mathcal{C} y conjunto de entrenamiento D .

- 1: $D' \leftarrow PreProcess(D)$
- 2: $k \leftarrow Selectk(\mathcal{C}, D')$
- 3: **return** D', k

Algoritmo para entrenar kNN tomado de [1, Figura 14.7]

Algoritmo 6 Apply- kNN

Parámetros de entrada: \mathcal{C}, D', k, d

- 1: $S_k \leftarrow ComputeNearestNeighbors(D', k, d)$
- 2: **for** $c_j \in \mathcal{C}$ **do**
- 3: $p_j \leftarrow |S_k \cap c_j|/k$
- 4: **return** $\arg \max_j p_j$

Algoritmo para aplicar kNN tomado de [1, Figura 14.7]

El parámetro k es usualmente seleccionado basado en el conocimiento que se posee sobre los problemas de clasificación similares al que se tiene. Otra forma de seleccionar k es usando conjuntos de documentos de prueba (ya clasificados) para ver qué valor de k producen mejores resultados.

También hay variantes del método donde lo que se hace es calcular una similitud entre el documento d a clasificar y cada uno de los k más cercanos, usando, por ejemplo, el coseno entre los vectores. Luego se hace un ranking entre las clases a las que pertenecen cada uno de los k puntos. Para ello se calcula para cada c , la suma de la similitud entre d y cada uno de los puntos que pertenecen a c y a la vez están entre los k mencionados. La siguiente función *score* produce

el resultado deseado

$$score(c, d) = \sum_{d' \in S_k(d)} I_c(d') \cos(\vec{v}(d'), \vec{v}(d)),$$

donde $S_k(d)$ es el conjunto de los k puntos más cercanos a d e $I_c(d')$ es 1 o 0 en dependencia de si d' pertenece a la clase c o no. Finalmente se selecciona para d la clase c que más alto aparezca en el ranking. En ocasiones esta variante presenta mayor exactitud que la anterior.

3.4. Medidas de evaluación

Una alta exactitud en la clasificación de los datos de entrenamiento no necesariamente se traduce en resultados correctos en los nuevos documentos introducidos. Puede suceder que el sistema resulte sobreentrenado (*over-fitting*) o subentrenado (*under-fitting*). El primero de estos ocurre cuando un sistema se entrena demasiado o se entrena con datos extraños. En este caso el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. Entonces el clasificador puede aprender incorrectamente a clasificar los elementos de alguna clase. Por otro lado *under-fitting* ocurre cuando el sistema tiene información muy vaga sobre la frontera entre clases lo que provoca cierta aleatoriedad en la clasificación.

Para evaluar la labor de un clasificador se debe emplear un conjunto de documentos diferentes al conjunto entrenante el cual se llama conjunto de prueba. Este debe estar en todo momento aislado del conjunto de entrenamiento. Si se juntan el clasificador puede aprender en su entrenamiento a reconocer los documentos del conjunto de prueba, y luego al evaluar el mismo, el desempeño del sistema será mucho más alto que el resultado real que tenga cuando se ponga en uso. Usualmente los datos que se tienen se dividen entre el conjunto entrenante y de prueba a razón de 4 : 1.

Las siguientes tres métricas pueden ser utilizadas para evaluar clasificadores de dos clases (c y \tilde{c}):

■ Precisión

$$Precision = \frac{\text{Número de documentos correctamente identificados como } c}{\text{Número de documentos identificados como } c}$$

■ Recobrado

$$recobrado = \frac{\text{Número de documentos correctamente identificados como } c}{\text{Número de documentos que pertenecen a } c}$$

■ Medida F

$$F = 2 \times \frac{Precision \times Recobrado}{Precision + Recobrado}$$

El mismo resultado se puede ver usando la matriz de contingencia. Una matriz de contingencia muestra de todas las pruebas realizadas con el clasificador, cuántas resultaron en verdadero positivo, falso positivo, falso negativo y verdadero negativo.

En el caso que la cantidad de clases sea mayor que dos, podemos computar promedios entre las métricas anteriores viendo el clasificador como uno de dos clases aplicado a cada clase c (y su complemento). Existen dos promedios que son los más usados para esto. Macropromedio calcula simplemente la media entre las medidas anteriores para cada clasificador de dos clases. Micropromedio primero adiciona las matrices de contingencia correspondientes a diferentes clases y luego aplica una de las métricas anteriores sobre la matriz resultante.

3.5. Ventajas y desventajas

Al igual que usar los métodos de agrupamiento tiene sus ventajas y desventajas, lo mismo sucede con los de clasificación. Nótese que, algunas de estas ventajas que se presentarán constituyen desventajas de los modelos no supervisados, y contrariamente, algunas de las ventajas de los modelos no supervisados constituyen desventajas de los supervisados. Esto es bueno porque si no es divisible con cuál de los métodos alinea mejor el problema a solucionar, con el análisis de pros y contras se puede reducir fácilmente la selección del correcto.

Se listan a continuación las ventajas y desventajas de la clasificación.

Ventajas:

- Al usar el aprendizaje supervisado se puede tener una idea exacta sobre las clases de los documentos, dado que estas se crean en base a características explícitas de los mismos.
- La salida suele ser más precisa que en el aprendizaje supervisado
- El proceso es fácil de comprender ya que las acciones de la máquina se reconocen con precisión.
- Es más fácil de trabajar con el aprendizaje supervisado que con el no supervisado.
- Se evalúa y recibe retroalimentación para comprobar la correctitud.

Desventajas:

- Se necesita un especialista que determine cuáles serán las clases en que se desea clasificar.
- Se requiere de un conjunto de datos entrenantes.
- Por lo general no se resuelven tareas tan complejas como en el aprendizaje supervisado.
- Los algoritmos solo funcionan dentro de las restricciones que se le han impuestos por lo que no proporcionan soluciones creativas.
- Suelen requerir mucho tiempo de entrenamiento.
- Puede predecir la salida incorrecta si los datos de prueba son diferentes de los datos de entrenamiento.

3.6. Aplicaciones en la Recuperación de la Información.

El aprendizaje de máquinas es ampliamente usado en la actualidad para resolver una variedad de problemas de muchas esferas. La clasificación, de conjunto con la regresión, son unas de las grandes tareas que tiene el aprendizaje de máquinas supervisado. Algunos problemas interesantes que resuelve la clasificación se encuentran dentro del marco de la Web y los Sistemas de Recuperación de la información. En el inicio de esta sección se presentó uno de ellos, que es la recuperación de consultas permanentes.

La organización del correo personal es otros de los ejemplos relacionados con la Web que pueden ser resueltos mediante clasificación. En muchas ocasiones una persona tiene un número grande de correos en el buzón de entrada. A la hora de revisarlo sería deseable que pudiera dirigirse directamente a aquellos que son de interés para él. Para ello la organización automática del correo en carpetas podría ser una ventaja invaluable. Un ejemplo particular sería la carpeta de correo spam.

Otro problema que se pudiera resolver sería la clasificación de valoraciones sobre algo en positivas o negativas. Esto tendría numerosas aplicaciones prácticas como pudiera ser la selección de una película. Un usuario podría revisar la cantidad de comentarios negativos, antes de lanzarse a disponer de dos horas de su tiempo viendo un material audiovisual que no resultará muy placentero. Los sistemas que se especializan en este tipo de sugerencias de contenido, son llamados usualmente sistemas de recomendación.

El uso más evidente que tiene la clasificación en la RI es la clasificación de documentos en tópicos. Esta clasificación facilitaría la implementación de un motor de búsqueda vertical, que permita al usuario restringir su búsqueda al tema deseado. Un SRI con semejante característica permite hacer una búsqueda más precisa en las consultas más especializadas por el usuario.

En RI es muy útil contar con un índice de los contenidos almacenados. En el proceso de creación del mencionado índice la clasificación puede jugar un importante papel. Por citar algunos ejemplos puede usarse para detectar el lenguaje de un documento, la segmentación y capitalización de las palabras y el codificado del documento.

3.7. Otros ejemplos de aplicación

Ya se habló en la epígrafe 3.6 acerca de la importancia de la clasificación en numerosos problemas de la sociedad. En especial se habló de algunos de los problemas específicos de la Recuperación de la Información en que es aplicable la clasificación. Se tratará en esta sección la importancia de la clasificación en problemas más generales de la humanidad.

La bioinformática es una rama científica que ha adquirido un desarrollo impresionante en los últimos años. Esta se dedica al desarrollo de herramientas informáticas para aplicarlas en la gestión y análisis de datos biológicos. Algunas de las aplicaciones más importantes que tiene la clasificación, y en general, el aprendizaje automático, es precisamente en esta rama de la ciencia. Con el uso

de sus algoritmos se permite automatizar la búsqueda de patrones en conjuntos de datos, que puedan ayudar a entender diferentes procesos biológicos que se manifiestan en los mismos. Debido a los grandes volúmenes de datos que se tienen ha crecido mucho en los últimos años la aplicación de la clasificación en esta disciplina.

Una de estas aplicaciones biológicas de las que hablabamos es la predicción de gen. Esta consiste en determinar que fracciones de una secuencia de ADN codifican proteínas. También se ve aplicado en la investigación de comunidades de microbios en nichos ecológicos. Así se puede obtener información taxonómica y metabólica de las comunidades estudiadas. Existen otras muchas aplicaciones en esta ciencia entre las que podemos mencionar la proteómica, los microarrays y la biología de sistemas.

Si se sale un tanto del marco de la bioinformática, pero no del todo lejos de la biología, se pueden encontrar algunas otras aplicaciones interesantes. Nos referimos al reconocimiento de algunos rasgos distintivos de una persona, como pueden ser, reconocimiento de rostros, huellas o voz. Este último lo tenemos al alcance de nuestras manos en asistentes virtuales como Siri o Google. Estas aplicaciones se entrenan para reconocer características distintivas de tu voz, mediante un proceso de aprendizaje supervisado, para luego poder diferenciar cuándo eres tú quien les está hablando.

El mercado financiero es una de los espacios de la vida del hombre donde más repercute la clasificación. Se puede emplear en mejorar el mercado digital, las ventas en línea, identificar el valor de vida útil de un cliente, la tasa de abandono, recomendaciones de productos y análisis de impacto de campañas de mercado.

En la seguridad informática también se evidencia la clasificación en acciones como detección de virus, enlaces maliciosos y fraude. Otras aplicaciones en que puede verse también son el Internet de las cosas (IoT), los motores de recomendación y la fijación de precios dinámicos de productos.

4. Conclusiones

La metodología usada en la clasificación y el agrupamiento es diferente y por ende el resultado esperado de sus algoritmos difiere también entre sí. Sin embargo, evidencian la creciente importancia que tiene la computación y en especial el aprendizaje de máquina, en ambas variantes, supervisado y no supervisado, para ayudar a la humanidad en tareas esenciales que se evidenciaban en el mundo incluso antes de su existencia. Hemos analizado aquí algunos de los algoritmos más importantes de ambas metodologías.

Aunque el objetivo de la clasificación y agrupamiento sea similar, hemos visto que esencialmente se utilizan en diferentes situaciones. Sus aplicaciones a la resolución de problemas se ven a menudo reflejados en la Recuperación de la Información y en la Web. Además, antes de aplicarlas en algún problema, es siempre bueno estar conscientes de sus ventajas y desventajas, las cuales fueron expuestas también en este trabajo.

Referencias

1. Maning C. D.: *An Introduction To Information Retrieval* (2009).
2. Berlin Chen: *Clustering Techniques for Information Retrieval* (2015).
3. Edie Rasmussen: *Information Retrieval* (2021).