

PRÁCTICA 2

DATA MINING

Inteligencia Artificial en las Organizaciones
Grado en Ingeniería Informática – Curso 2020/2021

Javier Cruz del Valle - 100383156
Gonzalo Fernández García – 100383212
Lucas González de Alba – 100383228



Índice

Introducción	4
Contexto.....	4
Planteamiento.....	5
Parte I - Desarrollo	5
Descripción de los datos utilizados	5
Descripción de los filtros creados y corpus generados	7
Solución final	13
Parte II - Desarrollo.....	16
Descripción del problema	16
Descripción de los modelos	16
Clasificación clústeres mediante árboles de decisión	17
Solución final	19
Descripción de resultados finales	20
Conclusión	20
Bibliografía	21

Introducción

Contexto

Llamamos minería de opinión o análisis de sentimiento al conjunto de técnicas de procesamiento del lenguaje natural y lingüística computacional dedicadas a la extracción de información subjetiva sobre un sujeto. En otras palabras, intentamos identificar el estado que mejor define a un perfil de usuario a través del análisis de sus expresiones, el campo semántico y vocabulario que utiliza.

En este proyecto el objetivo a grandes rasgos es estimar el grado de satisfacción de los usuarios basándonos en la similitud de su reseña al resto de comentarios de la aplicación para un lote dado de hoteles. Existen proyectos con planteamientos similares como el de *Berezina, K., Bilgihan, A., Cobanoglu, C., & Okumus, F* publicado en 2016. En este estudio 2510 hoteles fueron seleccionados de TripAdvisor.com para estimar en futuras reseñas el grado de pertenencia al grupo de clientes satisfechos e insatisfechos. En sus conclusiones destacaron que en promedio las reseñas de usuarios satisfechos remarcaban características intangibles de la estancia o la experiencia (atención al cliente, paisaje, atmosfera...), mientras que los clientes más descontentos se referían con mayor frecuencia a cualidades tangibles como el mobiliario, utensilios o transporte. De manera similar, aunque más general, la publicación de *Moghaddam, S., & Ester, M.* presenta un modelo llamado *Opinion Digger* que agrupa en orden ascendente de 1 a 5 el grado de satisfacción de los usuarios basándose en cualquier conjunto de reseñas. Otras propuestas aplican metodologías similares pero orientadas a categorizar el motivo o motivos por los que el cliente se ha desplazado al hotel. (*Xiang, Z., Du, Q., Ma, Y., & Fan, W. (2018)*).

A partir de esta literatura hemos construido una idea general del problema y procedido a resolverlo dadas las técnicas aprendidas en la asignatura.

Planteamiento

En el presente documento trabajaremos en el campo del Text Mining haciendo uso de una colección de datos semi-estructurados que más adelante comentaremos. El objetivo de la práctica puede fragmentarse en dos subobjetivos que pueden resumirse como la búsqueda del filtro óptimo, así como su corpus asociado a este y, en segundo lugar, la creación de distintos modelos mediante el algoritmo K-means para que posteriormente se seleccione la mejor versión y con esta misma se utilicen distintas generaciones de reglas y/o árboles de decisión que permitan completar el análisis mediante la descripción del cluster.

Respecto a los datos utilizados, haremos uso de un banco de datos del sitio web *Tripadvisor* que contiene una reseña escrita en inglés la cual está asociada a un valor comprendido entre [1,5]. Para poder trabajar con ello necesitaremos subdividir el contenido en cinco partes en función del valor de tal modo que se obtenga 500 instancias para cada uno de los subconjuntos. Una vez realizado se mezclarán todos ellos en un mismo archivo que servirá de corpus inicial para poder desarrollar la práctica. A su vez, el filtro inicial que nos permita establecer una comparativa básica será el que se encuentre por defecto en la herramienta empleada.

Parte I - Desarrollo

Descripción de los datos utilizados

En primer lugar, describiremos brevemente los datos analizando su composición interna, a diferencia de en la introducción, para saber qué buscamos.

Como ya hemos comentado se trata de un conjunto semi-estructurado sin orden alguno el cual se ha resumido en 2500 instancias que nos sirven para iniciar el proceso. Podemos observar que existe una gran cantidad de palabras coloquiales repetidas, así como números, pronombres, artículos, conjunciones que no aportan ningún tipo de conocimiento y por tanto serán nuestra mayor prioridad a la hora de establecer una métrica de mejora. Utilizando el filtro por defecto que establece Weka observamos lo siguiente:

No.	Name
1	<input type="checkbox"/> @@class@@
2	<input type="checkbox"/> 00
3	<input type="checkbox"/> 1
4	<input type="checkbox"/> 1/2
5	<input type="checkbox"/> 10
6	<input type="checkbox"/> 100
7	<input type="checkbox"/> 12
8	<input type="checkbox"/> 14
9	<input type="checkbox"/> 15
10	<input type="checkbox"/> 1st
11	<input type="checkbox"/> 2
12	<input type="checkbox"/> 20
13	<input type="checkbox"/> 2005
14	<input type="checkbox"/> 2006
15	<input type="checkbox"/> 2008
16	<input type="checkbox"/> 24

No.	Name
541	<input type="checkbox"/> large
542	<input type="checkbox"/> larger
543	<input type="checkbox"/> late
544	<input type="checkbox"/> later
545	<input type="checkbox"/> lawrence
546	<input type="checkbox"/> leave
547	<input type="checkbox"/> leaving
548	<input type="checkbox"/> left
549	<input type="checkbox"/> let
550	<input type="checkbox"/> level
551	<input type="checkbox"/> library
552	<input type="checkbox"/> light
553	<input type="checkbox"/> lighting
554	<input type="checkbox"/> lights
555	<input type="checkbox"/> like
556	<input type="checkbox"/> liked

No.	Name
1756	<input type="checkbox"/> quirky
1757	<input type="checkbox"/> renaissance
1758	<input type="checkbox"/> reviewer
1759	<input type="checkbox"/> satisfied
1760	<input type="checkbox"/> shampoo
1761	<input type="checkbox"/> slight
1762	<input type="checkbox"/> steep
1763	<input type="checkbox"/> stocked
1764	<input type="checkbox"/> sweet
1765	<input type="checkbox"/> traditional
1766	<input type="checkbox"/> upper
1767	<input type="checkbox"/> venice
1768	<input type="checkbox"/> vintage
1769	<input type="checkbox"/> warwick
1770	<input type="checkbox"/> worried
1771	<input type="checkbox"/> you

En efecto, no solo obtenemos números aleatorios, sino que debido a la redacción de los usuarios y las coincidencias que se generan los resultados contienen pronombres, artículos y palabras derivadas que no aportan nueva información.

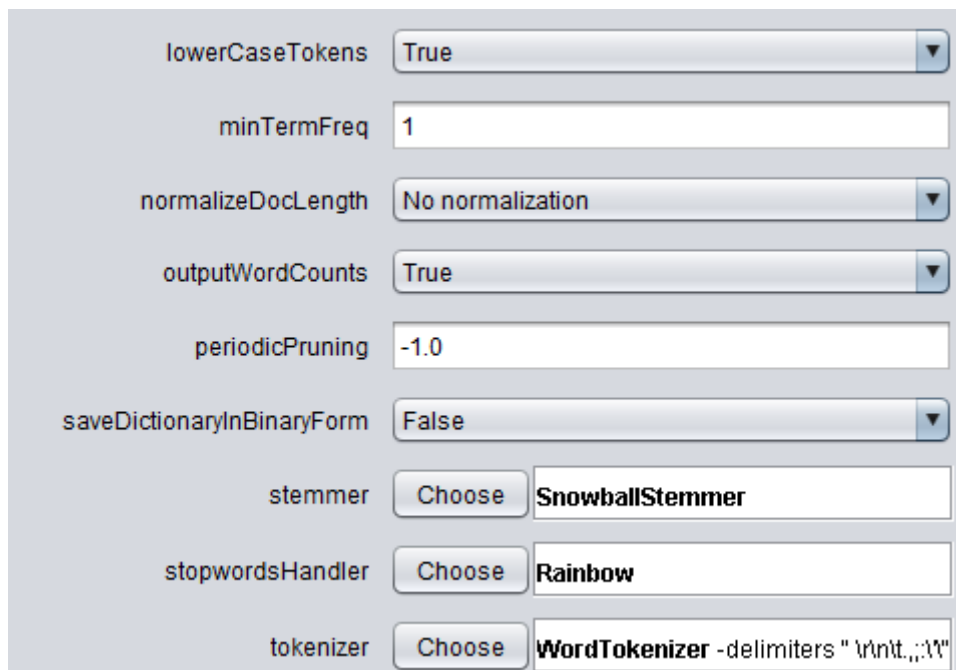
Establecido nuestro punto inicial comenzamos a comentar los filtros creados y sus corpus asociados respectivamente y, en último lugar, el filtro óptimo junto con su correcto corpus que hemos valorado como solución final.

Descripción de los filtros creados y corpus generados

A continuación, comentaremos un total de 4 filtros ordenados gradualmente de menor a mayor sofisticación del corpus que genera según las métricas que hemos planteado.

filtro1

En nuestro primer intento probamos a realizar un filtro que mantenga activadas las opciones que se marcan en el enunciado además de activar la opción que fuerza a que todos los tokens estén en minúsculas. Para las opciones del *stemmer*, *stopwordsHandler* y *tokenizer* escogeremos valores aleatorios para probar su funcionamiento y con ello obtenemos lo siguiente:



lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	Choose SnowballStemmer
stopwordsHandler	Choose Rainbow
tokenizer	Choose WordTokenizer -delimiters " \r\n\t.,;:!"

Cambios más importantes del filtro

No.	Name
1	<input type="checkbox"/> @@class@@
2	<input type="checkbox"/> 00
3	<input type="checkbox"/> 1
4	<input type="checkbox"/> 1/2
5	<input type="checkbox"/> 10
6	<input type="checkbox"/> 100
7	<input type="checkbox"/> 11
8	<input type="checkbox"/> 12
9	<input type="checkbox"/> 14
10	<input type="checkbox"/> 15
11	<input type="checkbox"/> 18
12	<input type="checkbox"/> 1st
13	<input type="checkbox"/> 2
14	<input type="checkbox"/> 20
15	<input type="checkbox"/> 2005
16	<input type="checkbox"/> 2006

No.	Name
1854	<input type="checkbox"/> season
1855	<input type="checkbox"/> secure
1856	<input type="checkbox"/> shampoo
1857	<input type="checkbox"/> slight
1858	<input type="checkbox"/> steep
1859	<input type="checkbox"/> stops
1860	<input type="checkbox"/> streetcar
1861	<input type="checkbox"/> tend
1862	<input type="checkbox"/> traditional
1863	<input type="checkbox"/> tuna
1864	<input type="checkbox"/> upper
1865	<input type="checkbox"/> vanity
1866	<input type="checkbox"/> venice
1867	<input type="checkbox"/> vintage
1868	<input type="checkbox"/> warwick
1869	<input type="checkbox"/> washington

Resultados más destacados

Con todo ello observamos que la única mejora se produce a la hora de eliminar ciertos pronombre, por lo que continuamos con la segunda parte.

filtro2

El segundo intento se basa en experimentar qué ocurre cuando el *stemmer* se pone a *nulStemmer* para poder deducir la importancia de este en base a los resultados que obtengamos. Junto con ello cambiamos poco más para poder sacar una buena conclusión de lo que hemos comentado por lo que tan solo variamos la asignación de otro *tokenizer*, obteniendo lo siguiente:

lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	Choose NullStemmer
stopwordsHandler	Choose Rainbow
tokenizer	Choose CharacterNGramTokenizer -max 3 -m

Cambios más importantes del filtro

No.	Name
1	<input checked="" type="checkbox"/> @@class@@
2	<input type="checkbox"/>
3	<input type="checkbox"/>
4	<input type="checkbox"/>
5	<input type="checkbox"/>
6	<input type="checkbox"/>
7	<input type="checkbox"/>
8	<input type="checkbox"/>
9	<input type="checkbox"/>
10	<input type="checkbox"/> 1
11	<input type="checkbox"/> 2
12	<input type="checkbox"/> 3
13	<input type="checkbox"/> 5
14	<input type="checkbox"/> ac
15	<input type="checkbox"/> ad
16	<input type="checkbox"/> ar

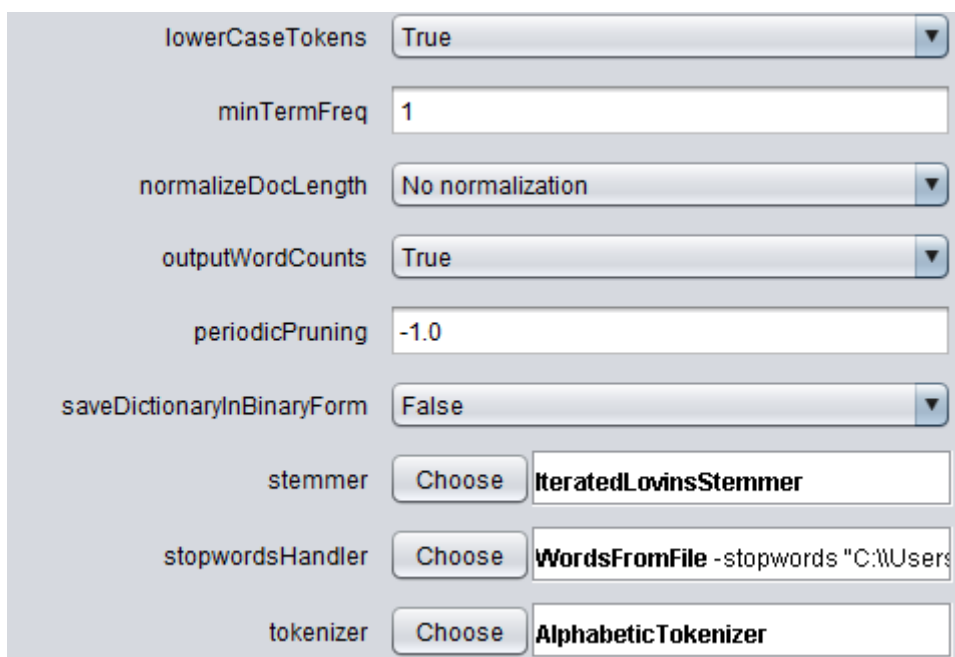
No.	Name
1211	<input type="checkbox"/> lki
1212	<input type="checkbox"/> llo
1213	<input type="checkbox"/> ly,
1214	<input type="checkbox"/> n g
1215	<input type="checkbox"/> n l
1216	<input type="checkbox"/> noi
1217	<input type="checkbox"/> ntr
1218	<input type="checkbox"/> nve
1219	<input type="checkbox"/> obb
1220	<input type="checkbox"/> onv
1221	<input type="checkbox"/> opp
1222	<input type="checkbox"/> oye
1223	<input type="checkbox"/> ppi
1224	<input type="checkbox"/> tm
1225	<input type="checkbox"/> uie
1226	<input type="checkbox"/> wnt

Resultados más destacados

Finalmente concluimos con la importancia de un buen *stemmer* debido a que sin este los resultados son completamente caóticos. Punto el cual nos centraremos en la próxima iteración.

filtro3

Tras observar los pésimos resultados anteriores nos centraremos en obtener un buen *stemmer* que nos permita la eliminación de las primeras instancias de los anteriores corpus compuestas por números aleatorios e incluso instancias vacías. Para ello modificamos los siguientes valores: *stemmer*, *stopwordsHandler* y *tokenizer*. Tanto el *stemmer* como el *tokenizer* son fruto de distintos intentos intermedios hasta dar con la combinación correcta, sin embargo, el *stopwordsHandler* se escoge como recurso dado en el enunciado de la práctica. Con todo ello obtenemos lo siguiente:



lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	Choose IteratedLovinsStemmer
stopwordsHandler	Choose WordsFromFile -stopwords "C:\Users
tokenizer	Choose AlphabeticTokenizer

Cambios más importantes del filtro

No.	Name
1	<input type="checkbox"/> @@class@@
2	<input type="checkbox"/> a
3	<input type="checkbox"/> abl
4	<input type="checkbox"/> absolut
5	<input type="checkbox"/> ac
6	<input type="checkbox"/> acc
7	<input type="checkbox"/> accommod
8	<input type="checkbox"/> accomod
9	<input type="checkbox"/> act
10	<input type="checkbox"/> actu
11	<input type="checkbox"/> ad
12	<input type="checkbox"/> addit
13	<input type="checkbox"/> adequ
14	<input type="checkbox"/> adjac
15	<input type="checkbox"/> adv
16	<input type="checkbox"/> afford

No.	Name
1584	<input type="checkbox"/> ship
1585	<input type="checkbox"/> signif
1586	<input type="checkbox"/> sleeper
1587	<input type="checkbox"/> soundproof
1588	<input type="checkbox"/> steep
1589	<input type="checkbox"/> streetcar
1590	<input type="checkbox"/> temp
1591	<input type="checkbox"/> treadmill
1592	<input type="checkbox"/> tun
1593	<input type="checkbox"/> upper
1594	<input type="checkbox"/> util
1595	<input type="checkbox"/> ven
1596	<input type="checkbox"/> vict
1597	<input type="checkbox"/> vint
1598	<input type="checkbox"/> warwick
1599	<input type="checkbox"/> washingt

Resultados más destacados

En este tercer paso observamos una increíble mejora ya que no solo somos capaces de eliminar los números, sino que reducimos considerablemente el número de instancias respecto al inicio (la segunda iteración no la contemplamos) aunque siendo objetivos los resultados están lejos de ser óptimos ya que no solo hay derivaciones, sino que se generan letras sueltas carentes de significado. Por lo que tenemos un nuevo objetivo para la próxima (y última) iteración experimental.

filtro4

En último lugar, afinamos los parámetros para acercarnos a un filtro capaz de eliminar los números, derivaciones, pronombres, letras sueltas y cualquier otra instancia que no aporte conocimiento alguno.

Haciendo uso del filtro anterior observamos que el *tokenizer* ha funcionado correctamente por lo que dejamos el mismo, sin embargo, pensamos que tal vez combinando distintos *stopwordsHandler* obtengamos una mejor solución y que su *stemmer* sea una pequeña variación del utilizado en la iteración previa. Con ello obtenemos el siguiente corpus quasi-óptimo:



lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	Choose LovinsStemmer
stopwordsHandler	Choose MultiStopwords -stopwords weka.core.
tokenizer	Choose AlphabeticTokenizer

Cambios más importantes del filtro

No.	Name
1	<input checked="" type="checkbox"/> @@class@@
2	<input type="checkbox"/> abl
3	<input type="checkbox"/> absolut
4	<input type="checkbox"/> ac
5	<input type="checkbox"/> accept
6	<input type="checkbox"/> acces
7	<input type="checkbox"/> accommod
8	<input type="checkbox"/> accomod
9	<input type="checkbox"/> accur
10	<input type="checkbox"/> act
11	<input type="checkbox"/> actu
12	<input type="checkbox"/> ad
13	<input type="checkbox"/> addit
14	<input type="checkbox"/> adequ
15	<input type="checkbox"/> adjac
16	<input type="checkbox"/> advant

No.		Name
1706	<input type="checkbox"/>	system
1707	<input type="checkbox"/>	temp
1708	<input type="checkbox"/>	thread
1709	<input type="checkbox"/>	tid
1710	<input type="checkbox"/>	transit
1711	<input type="checkbox"/>	treadmil
1712	<input type="checkbox"/>	tun
1713	<input type="checkbox"/>	upper
1714	<input type="checkbox"/>	util
1715	<input type="checkbox"/>	vancouver
1716	<input type="checkbox"/>	venic
1717	<input type="checkbox"/>	victor
1718	<input type="checkbox"/>	vint
1719	<input type="checkbox"/>	warwick
1720	<input type="checkbox"/>	washingt
1721	<input type="checkbox"/>	zo

Resultados más destacados

Sin duda alguna esta última iteración, previa al modelo final, es un completo éxito ya que eliminamos una gran cantidad de derivaciones y aumentamos el número de instancias siendo un corpus más heterogéneo debido a la limpieza del *stemmer*. Sin embargo, parece ser inevitable obtener palabras compuestas por letras sueltas que no aportan realmente mucha información más allá de alguna abreviatura por lo que no contesto con ello decidimos forzar un poco más los parámetros para dar con el filtro y corpus perfectos.

Solución final

filtro5

Después de varios intentos obtenemos el que consideramos el mejor filtro debido a su capacidad para no solo eliminar incoherencias y números, sino que es capaz de filtrar redundancias, descartar pronombres y artículos, y resumir derivaciones en una solo instancia. Los parámetros utilizados son muy similares a la última iteración experimental con la única gran diferencia de usar el *stemmer* inicial. Este es el resultado final:

lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	Choose SnowballStemmer
stopwordsHandler	Choose MultiStopwords -stopwords "weka.co
tokenizer	Choose AlphabeticTokenizer

Cambios más importantes del filtro

No.	Name
1	<input checked="" type="checkbox"/> @@class@@
2	<input type="checkbox"/> absolute
3	<input type="checkbox"/> absolutely
4	<input type="checkbox"/> access
5	<input type="checkbox"/> accommodating
6	<input type="checkbox"/> accomodate
7	<input type="checkbox"/> accomodating
8	<input type="checkbox"/> accurate
9	<input type="checkbox"/> ace
10	<input type="checkbox"/> add
11	<input type="checkbox"/> added
12	<input type="checkbox"/> additional
13	<input type="checkbox"/> adequate
14	<input type="checkbox"/> adjacent
15	<input type="checkbox"/> advantage
16	<input type="checkbox"/> advisor

No.	Name
109	<input type="checkbox"/> booking
110	<input type="checkbox"/> boston
111	<input type="checkbox"/> bothered
112	<input type="checkbox"/> bottle
113	<input type="checkbox"/> bottled
114	<input type="checkbox"/> bought
115	<input type="checkbox"/> bourbon
116	<input type="checkbox"/> boutique
117	<input type="checkbox"/> brand
118	<input type="checkbox"/> break
119	<input type="checkbox"/> breakfast
120	<input type="checkbox"/> breakfasts
121	<input type="checkbox"/> bright
122	<input type="checkbox"/> brilliant
123	<input type="checkbox"/> bring
124	<input type="checkbox"/> Broadway

No.	Name
1012	<input type="checkbox"/> theatre
1013	<input type="checkbox"/> theatres
1014	<input type="checkbox"/> theme
1015	<input type="checkbox"/> thing
1016	<input type="checkbox"/> things
1017	<input type="checkbox"/> thought
1018	<input type="checkbox"/> throw
1019	<input type="checkbox"/> tickets
1020	<input type="checkbox"/> time
1021	<input type="checkbox"/> times
1022	<input type="checkbox"/> tiny
1023	<input type="checkbox"/> tips
1024	<input type="checkbox"/> tired
1025	<input type="checkbox"/> toast
1026	<input type="checkbox"/> toilet
1027	<input type="checkbox"/> toiletries

No.	Name
1444	<input type="checkbox"/> supposed
1445	<input type="checkbox"/> swim
1446	<input type="checkbox"/> swimming
1447	<input type="checkbox"/> talked
1448	<input type="checkbox"/> taste
1449	<input type="checkbox"/> tax
1450	<input type="checkbox"/> telephone
1451	<input type="checkbox"/> temperature
1452	<input type="checkbox"/> terrible
1453	<input type="checkbox"/> thankfully
1454	<input type="checkbox"/> thinking
1455	<input type="checkbox"/> tip
1456	<input type="checkbox"/> tipped
1457	<input type="checkbox"/> tours
1458	<input type="checkbox"/> towel
1459	<input type="checkbox"/> traveled

No.	Name
1866	<input type="checkbox"/> satisfied
1867	<input type="checkbox"/> secure
1868	<input type="checkbox"/> shampoo
1869	<input type="checkbox"/> slight
1870	<input type="checkbox"/> steep
1871	<input type="checkbox"/> stops
1872	<input type="checkbox"/> streetcar
1873	<input type="checkbox"/> tap
1874	<input type="checkbox"/> tend
1875	<input type="checkbox"/> tuna
1876	<input type="checkbox"/> upper
1877	<input type="checkbox"/> vanity
1878	<input type="checkbox"/> venice
1879	<input type="checkbox"/> vintage
1880	<input type="checkbox"/> warwick
1881	<input type="checkbox"/> washington

Resultados más destacados

Concluimos la primera parte dando paso a la sección de clústeres donde utilizaremos los resultados finales que hemos analizado.

Parte II - Desarrollo

Descripción del problema

Una vez que hemos obtenido el corpus mediante el mejor de los filtros aplicados anteriormente, se ha procedido a crear diez modelos de clúster diferentes mediante el algoritmo de clusterización simpleKmeans, en los distintos modelos se han variado 3 parámetros:

- Número de clústeres: este valor indica la cantidad de clústeres que se van a generar. Se han probado diferentes modelos en los que el valor del parámetro va desde pequeños como 2 hasta grandes como 10 pasando por cantidades intermedias.
- La función de distancia: esta función determina como se van a comparar las diferentes instancias. Se ha variado en las dos alternativas válidas en Weka para este problema que son la distancia manhattan y la distancia euclídea.
- El número de semillas: este valor indica el número de semillas aleatorias usadas para construir los clusters. Para él se han probado diferentes cantidades desde tan solo 1 hasta 1000.

Descripción de los modelos

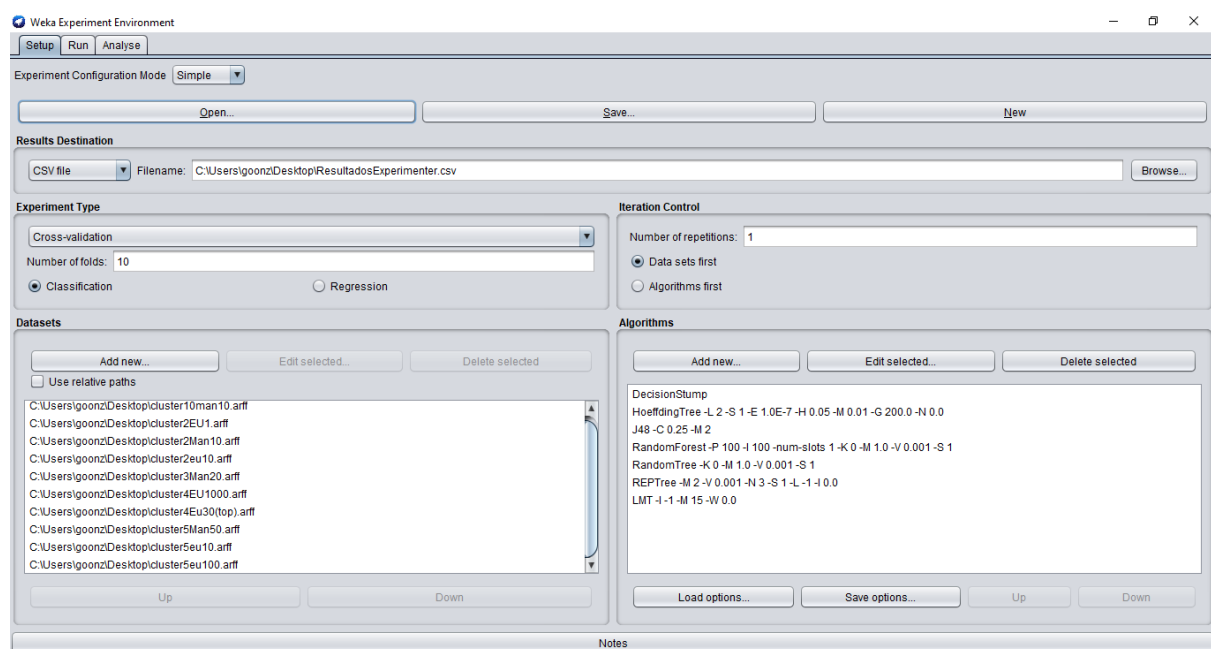
A continuación, se muestra una tabla con las distintos modelos que se han creado:

Numero de clústeres	Función de distancia	Función de semillas
2	Euclídea	100
2	Euclídea	10
2	Manhattan	10
3	Manhattan	20
4	Euclídea	30
4	Euclídea	1000
5	Euclídea	10
5	Euclídea	100
5	Manhattan	50
10	Manhattan	10

El siguiente paso ha sido guardar los resultados de estos procesos como un archivo .arff y eliminar el atributo 'Instance_number' para poder realizar la clasificación mediante la construcción de diferentes árboles de decisión.

Los nombres de los documentos tienen el siguiente código: el primer número es el número de clústeres, el siguiente elemento es la función de distancia y el tercer número es el de semillas aleatorias.

Se ha utilizado la aplicación 'experimenter' de Weka para hacer pruebas con los 10 ficheros resultantes de los clusters y los siete árboles de clasificación que permite realizar Weka sobre este problema. Por tanto, se ha aplicado sobre cada uno de los conjuntos los siete algoritmos de árboles con un cross-validation de 10 capas. Los siete algoritmos de árboles de decisión son: DecisionStump, HoeffdingTree, J48, LMT, RandomForest, RandomTree y REPTree.



Experimento ejecutado

Clasificación clústeres mediante árboles de decisión

Una vez obtenido estos resultados hemos realizado un estudio para analizar las diferentes clusterizaciones realizadas por las distintas configuraciones del algoritmo Kmeans y los resultados de los distintos árboles.

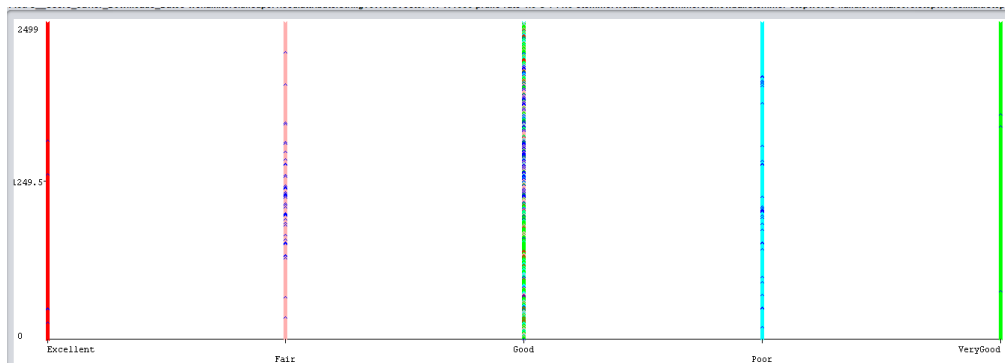
Estas son las conclusiones que hemos obtenido tras el análisis de los resultados:

- En aquellos modelos con poca cantidad de clusters se ha observado que son poco interesantes ya que lo más óptimo es generar 4 o 5 clusters distintos al tener 5 diferentes tipos de valoraciones.
- Al igual que con los modelos de pocos clusters en aquellos con gran cantidad se han decidido descartar por la misma razón que en el punto anterior se ha expuesto.
- Se ha observado que hay muchos modelos en los que la repartición de las instancias en los clusters no es muy equitativa existiendo cluster con 2 instancias y otro con miles en estos casos también se han descartado ya que las instancias deben estar más o menos repartidas equitativamente al tener la misma cantidad de instancias de cada valoración.
- En general se ha observado que son mejores los modelos generados con distancia euclídea que con distancia manhattan en igualdad para el resto de los parámetros del algoritmo kmeans. Esto se debe a que con distancia euclidiana se reparten mejor las instancias y además el porcentaje de instancias clasificadas correctamente por los árboles de decisión es mayor.
- Por parte del número de semillas aleatoria se ha observado que para el modelo en el que era muy elevado (1000) los resultados han sido muy malos al no repartir bien la instancias pese a ser cinco clusters que es uno de los valores óptimos.
- Al igual que con el número de semillas muy alto cuando este valor ha sido de 1 los resultados han sido bastante malos al no repartir equitativamente las instancias.
- En cuanto al tipo de árbol los mejores resultados en general se han obtenido para el algoritmo LMT, aunque el algoritmo J48 también ha generado buenos resultados. Los algoritmos que peor resultado de instancias clasificadas correctamente han sido RandomForest y RandomTree.

Solución final

Tras realizar todo este análisis se ha concluido que los mejores resultados se han obtenido para el cluster generado con los siguientes parámetros:

- Número de clústeres: 5
- Función de distancia: euclídea
- Número de semillas: 100



Observamos como la clasificación por término es mejor cuanto más “radical” (‘Excelente’, ‘pobre’, o ‘muy bueno’) es el estado o sentimiento que motiva la reseña. Esto se explica gracias a que con cuanto menos ambiguo sea este, menos términos del corpus compartirá la reseña y mejor se clasifica. Por tanto, vemos como ‘Bueno’ y ‘decente’ incorporan varios de los clústeres generados.

Además, el árbol de predicción que se ha decidido usar ha sido el LMT que obtiene los resultados que se muestran a continuación:

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      2334           93.36 %
Incorrectly Classified Instances    166            6.64 %
Kappa statistic                    0.9145
Mean absolute error                 0.0331
Root mean squared error             0.14
Relative absolute error             10.6255 %
Root relative squared error         35.4843 %
Total Number of Instances          2500

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      0,784    0,008    0,892    0,784    0,835     0,824    0,990    0,917    cluster0
      0,948    0,006    0,976    0,948    0,962     0,953    0,998    0,994    cluster1
      0,965    0,032    0,919    0,965    0,941     0,919    0,996    0,991    cluster2
      0,953    0,019    0,933    0,953    0,943     0,927    0,997    0,991    cluster3
      0,914    0,021    0,927    0,914    0,921     0,897    0,992    0,974    cluster4
Weighted Avg.    0,934    0,019    0,934    0,934    0,933     0,916    0,996    0,982

=== Confusion Matrix ===

  a  b  c  d  e  <-- classified as
149  1  6  9  25 | a = cluster0
 1496 20  4  2  | b = cluster1
  1  6 655  9  8  | c = cluster2
  2  3 14 512  6  | d = cluster3
 14  2 18 15 522  | e = cluster4

```

Descripción de resultados finales

De acuerdo al desarrollo siguiente desarrollo:

1. Selección del mejor filtro (*stopwords*, *stemmer*, *tokenizer*) y construcción del Corpus
2. Construcción de clúster (número de clústeres, distancia, semilla) mediante k-medias y sus análisis con árboles de decisión disponibles.

Seleccionamos el modelo con mejores resultados Clúster (número de clústeres = 5, distancia Euclídea = 10, semillas = 10) y LMT ya que sus clústeres asemejan con mayor precisión (menor porcentaje de instancias mal clasificadas) las cinco categorías de satisfacción del usuario.

Conclusión

El proyecto ha servido como punta de lanza para adentrarnos en el campo de la **minería de datos** y ha sido gracias al estudio del proyecto que se ha arrojado algo de luz sobre el campo de **análisis de sentimiento**, un área hasta ahora desconocida por nosotros. Cabe destacar que gracias a la primera práctica de la asignatura (Aprendizaje Automático) nos hemos visto mejor preparados para enfrentarnos al reto, ya que a nivel conceptual hemos interiorizado la metodología de preprocesado de datos, configuración de modelos y sistematización de prueba, error y ajuste típicas de la inteligencia artificial. No obstante, habría que destacar una diferencia clave entre ambas y es que en la primera se utilizó una **red neuronal supervisada** sobre datos **estructurados** para predecir **valores** de una **regresión** mientras que en esta se trata de **procesar** datos **no estructurados** (construyendo un vector de frecuencias ponderadas, un **corpus**, de manera **no supervisada**) con el objetivo de **clasificarlos** mediante **clústeres** y algoritmos **de generación de reglas y/o árboles de decisión**. Observamos ciertos paralelismos, por ejemplo, cuando se escogían hiperparámetros "ciegamente", en este caso se seleccionaban criterios de filtrado. En cualquier caso, para este proyecto el preprocesado de datos se ha llevado una parte importante del tiempo de trabajo.

Para terminar, creemos relevante citar la publicación de Weismayer, C., Pezenka, I., & Gan, C. H. K. (2018), que resume muy bien los avances en la minería de datos subjetivos (opinión mining) y recoge una conclusión muy valiosa de cara al futuro de este campo. Los investigadores de este proyecto concluyeron que las técnicas de procesamiento del lenguaje natural y lingüística computacional permiten resolver problemas que la metodología tradicional no alcanza a resolver y que en el futuro el análisis de sentimiento irá íntimamente ligado a los sistemas que gestionen contenido generado por el usuario.

Por tanto, como conclusión, creemos que el proyecto que se nos ha encomendado es tanto académico como útil ya que es representativo de algunos de los problemas reales con publicaciones y líneas de investigación.

Bibliografía

1. Weismayer, C., Pezenka, I., & Gan, C. H. K. (2018). Aspect-based sentiment detection: Comparing human versus automated classifications of TripAdvisor reviews. In *Information and Communication Technologies in Tourism 2018* (pp. 365-380). Springer, Cham.
2. Berezina, K., Bilgihan, A., Cobanoglu, C., & Okumus, F. (2016). Understanding satisfied and dissatisfied hotel customers: text mining of online hotel reviews. *Journal of Hospitality Marketing & Management*, 25(1), 1-24.
3. Xiang, Z., Du, Q., Ma, Y., & Fan, W. (2018). Assessing reliability of social media data: lessons from mining TripAdvisor hotel reviews. *Information Technology & Tourism*, 18(1-4), 43-59.
4. Moghaddam, S., & Ester, M. (2010, October). Opinion digger: an unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1825-1828).