Proyecto de Minería de Datos:
Borrador de memoria completa

Markus Fischer • Guzmán López • David Pérez • Ander Raso

Índice

1. Introducción

1.1. Objetivo de la tarea

En este proyecto se nos encarga la tarea de trabajar en el campo del *Text Mining* y del clustering de documentos. Nuestro objetivo es, a partir de una gran colección de textos, buscar formas de agruparlos y tratar de extraer conclusiones de los resultados que obtengamos.

1.2. Propuesta de trabajo del grupo

Como grupo, hemos decidido realizar la tarea sobre la colección de autopsias verbales que se propuso como opción. Esto se debe a que, ademas de considerar el tema muy interesante, creemos que una propuesta que se encuentra muy próxima al uso que se da al *Text Mining* en el ámbito científico.

Sin embargo, de esta decisión también surgen ciertos retos a los que debemos poner solución:

- Muchos de los reportes de autopsia están en un lenguaje poco preciso y muchas veces ininteligible, probablemente causado tanto por el desconocimiento de los que dieron el reporte, como por las traducciones que se han hecho a estos.
- En muchas ocasiones no hay reporte verbal o este es irrelevante, por lo que la única información útil de que se dispone es de los datos del difunto, tales como país, edad, sexo, etc.
- TODO: Añadir alguna más para que no quede vacío.

2. Descripción y análisis de datos

En este proyecto vamos a trabajar con autopsias verbales. Éstas son reportes que dan familiares o personas cercanas a un fallecido en lugares donde por norma general no se realiza una autopsia post-mortem a menos que sea estrictamente necesario. Estas autopsias se componen de información básica del paciente (edad, sexo, etc) así como del reporte oral que ha dado el relativo al que se ha entrevistado.

La base de datos de las autopsias verbales se compone de casi 12000 instancias. En cada instancia disponemos de una serie de atributos:

- **newid:** Identificador númerico de la instancia.
- **module:** Grupo de edad del fallecido. Los valores posibles son “neonate”, “child” y “adult”.
- **site:** Lugar del que proviene la autopsia: los valores posibles para este campo son:
  - AP=Andhra Pradesh, India
  - Dar=Dar es Salaam, Tanzania

- UP=Uttar Pradesh, India
- Pemba=Pemba, Tanzania
- Bohol=Bohol, Philippines
- Mexico=Distrito Federal, Mexico

- **gs_text34:** Causa de la muerte del paciente.
- **sex:** Sexo del difunto. Los valores posibles son 1, 2, 3 y 4. Cada uno de estos valores hace referencia a "hombre", "mujer", "se niega a responder" y "desconocido" respectivamente.
- **age_years:** Edad del fallecido en años. En caso de ser igual a 999 significa que es desconocida.
- **age_months:** Edad del fallecido en meses. En caso de ser igual a 99 significa que es desconocida.
- **age_days:** Edad del fallecido en días. En caso de ser igual a 99 significa que es desconocida.
- **open_response:** Declaración de la persona cercana al difunto en caso de que hubiese una. En este texto se han eliminado todas las referencias que se diesen en la declaración que pudiesen relacionarse con el difunto para asegurar su privacidad, tales como menciones a los lugares y hospitales en los que se ha encontrado, el nombre de personas o fechas concretas en relación al fallecido, etc. En su lugar se han sustituído por palabras neutrales entre corchetes, tales como "[PERSON]" o "[HOSPITAL]".

3. Preproceso de datos

El formato original del archivo de autopsias es un xlsx, es decir, una hoja de cálculo de Microsoft Office. Esto nos facilita mucho la tarea de procesar los textos, ya que es muy fácil convertir de xlsx a formatos de texto plano. En nuestro caso elegimos csv, ya que es un formato muy limpio y que más adelante nos será útil para transformarlo al formato arff, el cual es compatible con Weka y gran parte de las librerias de Data Mining que hemos utilizado.

Una vez tenemos nuestro documento como csv, procedemos a preprocesarlo. Ya que los valores identificador, grupo de edad, lugar, diagnóstico, sexo y edad están codificados segun unos criterios que se mantienen a lo largo de todas las instancias, sabemos que no debemos preprocesarlos. Sin embargo, es en el campo de la respuesta verbal en el que tenemos que solucionar algunas anomalías:

- **Mayúsculas y minúsculas:** Para evitar diferencias entre palabras escritas completamente en mínuscula, completamente en mayúscula o con la primera letra mayúscula, debemos transformar todas ellas a un mismo formato. En nuestro caso el formato escogido son las minúsculas.
- **Saltos de línea:** En algunos casos nos encontramos con saltos de línea internos en el texto. Ya que csv separa sus diferentes atributos por comas y

las instancias por líneas, los saltos de línea internos en el texto crean inconsistencias y errores al procesar el csv. El método más fácil para desacerse de ellos pero mantener el texto en el mismo formato es sustituírlos por espacios.

- **Números:** Los números, pese a ser uno de los elementos más comunes en los textos, han sido uno de los más díficiles de preprocesar, no por su dificultad en cuanto a programar, sino a la decisión que tomar en cuanto a ellos. Esto se debe a que se toman como una palabra por si mismos y pese a tener un peso informativo alto, sólo es así cuando va acompañado de otra palabra a la que calificar. Ya que los programas de text mining los van a considerar independientes del resto de palabras, hemos considerado que la información que van a aportar es suficientemente baja como para simplemente eliminarlos del texto.

- **Tokens:** Se consideran tokens a todas las palabras recurrentes que no tienen ni significado ni peso informativo, como por ejemplo "the" o "a". Mediante un diccionario de tokens en la lengua inglesa hemos podido detectar todos ellos y eliminarlos del texto.

- **Símbolos:** Aquí tenemos varios problemas que debemos arreglar de diferentes formas.

  - **Barras "/":** En ocasiones nos encontramos con dos palabras escritas de modo "palabra1/palabra2". Para evitar que se procesen ambas como una misma palabra, realizamos el mismo procedimiento que utilizamos con los saltos de línea: cambiamos las barras por espacios.
  - **Corchetes "[ ]":** En gran parte de las instancias que disponen de autopsia verbal nos encontramos con referencias a nombres de personas, hospitales, años concretos, etc. Para preservar la privacidad de los fallecidos y sus relativos estos han sido sustituídos por sus correspondientes palabras clave entre corchetes, como por ejemplo "[PATIENT]" o "[HOSPITAL]". Ya que estos datos son en la inmensa mayoría de casos de poca utilidad, hemos decidido que estás palabras serán completamente eliminadas de los textos.

Después de limpiar los textos de las instancias necesitamos convertir ese texto a un formato que podamos utilizar. El formato escogido es el TF-IDF, el cual consiste en una representación vectorial del conjunto de palabras presentes en todos los documentos (instancias). Cada una de las palabras se evalúa individualmente y se le asigna un valor numérico en función al número de apariciones que tiene en cada documento, pero teniendo en cuenta también en cuántos de los documentos aparece para evitar darle excesiva importancia a palabras que simplemente se repiten por ser comunes.

Escogimos la representación TF-IDF en favor de *Bag of Words* por el hecho de que aporta más información. *Bag of Words* se limita a inicar la presencia o ausencia de palabras en los documentos mientras que TF-IDF le asigna un valor en función de su aparente importancia teniendo en cuenta todos los documentos.

4. Clustering

Para el clustering hemos decidido utilizar el algoritmo K-Means, el cual implementaremos en Java.

blah, blah, blah. no se que poner aqúi. brb.

Una vez que tenemos las instancias separadas en clusters, sería interesante representarlas gráficamente para poder ver como se relacionan los clusters, calculados a partir del texto describiendo la enfermedad, con la causa de la muerte anotada en las autopsias. Para ello, tenemos que reducir el enorme espacio de atributos de las instancias a algo que podamos representar en un espacio de no más de tres dimensiones. Para ello pensamos en utilizar el método PCA (*Principal Components*) que hace precisamente eso: reduce un gran número de variables a una serie de valores representativos, ortogonales entre sí para representarlos en espacios de pocas dimensiones.
Sin embargo nos encontramos con un problema: el PCA es un método costoso, tanto en tiempo como en uso de memoria. Al tratar de aplicar el PCA a nunestro conjunto de casi 12000 instancias, y teniendo alrededor de 8000 atributos en cada una de ellas (a causa del formato TF-IDF) el programa no fue capaz de terminar el proceso, siempre resultando en una falta de memoria, sin importar le permitiesemos utilizar.
Pensamos en llevar a cabo el PCA individualmente en cada clusters, dividiendo la carga en distintos intentos, pero como el PCA utiliza todo el conjunto para determinar los componentes principales, creemos que realizar el proceso en cada cluster de forma separada en lugar de sobre todo el conjunto podría causa inconsistencias a la hora de reducir los vectores.