CLASIFICACIÓN DE TEXTO EN LENGUAJE NATURAL

Inteligencia artificial avanzada

Cristina Garrido Amador
Jesús Ramos Álvarez

Índice

Introducción

El objetivo a cumplir es construir un sistema para la clasificación de mensajes que emiten las empresas en las redes sociales, de manera que se puedan dividir en informativos (afirmaciones objetivas sobre la empresa o actividades), diálogo (respuestas a usuarios, etc.) o acción (mensajes que piden votos, que entren en un enlace, etc.).

Creación de los corpus

La primera tarea que se pide para la realización del objetivo final es crear unos ficheros corpus que contengan los mensajes de las empresas, de forma que existan tres de ellos, uno para información, otro para diálogo y otro para acción.

Antes de realizar la separación, lo primero que se tuvo que hacer fue limpiar el fichero, es decir, eliminar caracteres raros que al copiar de la plataforma se copiaron mal o sustituirlos por el carácter que era realmente.

Una vez realizada la limpieza, se pasa a crear los corpus. En esta nueva tarea lo que se hizo fue separar el fichero inicial en tres, de forma que, se fueron cogiendo todos los tweets de un tipo, copiando a un fichero .txt y guardándolos con el nombre correspondiente, es decir, si los mensajes a los que hacía referencia eran de información, el fichero se llamaría corpusI.txt, de esta forma se tendrían los otros dos que se llaman corpusD.txt y corpusA.txt.

A continuación, se pedía tener un corpus que tuviese guardados todos los mensajes una vez hecha la limpieza, de forma que, se copiaron los corpus comentados con anterioridad en uno solo llamándolo corpustodo.txt.

Creación del vocabulario

Seguidamente, después de haber creado los distintos corpus, se pasa a crear un programa que saque el vocabulario, es decir, lo que se hará será crear un programa que dado un corpus saque un vocabulario, en el cual las palabras deben aparecer alfabéticamente.

Para realizar este programa se utilizó el lenguaje Java dado que es más versátil y más cómodo de implementar. Lo primero que se hizo, una vez leído el fichero fue ir analizando las líneas de manera que si encontraba un símbolo como @ o # eliminase la palabra que le seguía dado que eso no aportaba información al vocabulario. Por otro lado, si encontraba la palabra http también eliminaba lo que le seguía, pues una URL tampoco servía para el vocabulario.

Mientras tanto, iba leyendo las palabras y las iba guardando en un fichero, de forma que si esa palabra ya existe no la guardaba en el vocabulario.

Finalmente, una vez acabado de leer completamente el fichero del corpus y una vez creado el vocabulario, se pasó a ordenar las palabras para que se quedasen en orden alfabético.

Estimación de probabilidades

Finalizada la parte del vocabulario, se pasa a estimar las probabilidades de las palabras. Para ello, se pide calcular la frecuencia de una palabra en el corpus y la estimación del logaritmo de su probabilidad, donde se utilizará el suavizado laplaciano con tratamiento de palabras desconocidas. Este fichero, también deberá estar ordenado alfabéticamente.

Para llevar a cabo esta nueva tarea, se realizará por corpus de forma que, al programa se le de como entrada el vocabulario y el corpus, por ejemplo, el de información.

A continuación, por cada nueva palabra que encuentra cuenta cuantas veces sale en ese mismo corpus y lo apunta en un nuevo fichero llamado Aprendizaje.txt. En este fichero, se guardará la palabra, la frecuencia de esta y el logaritmo de su probabilidad.

La fórmula que se utiliza para calcular el logaritmo de la probabilidad de cada palabra es la siguiente:

$$\log\left(\frac{numero\ apariciones\ de\ la\ palabra\ en\ el\ corpus + 1}{numero\ palabras\ del\ corpus\ +\ tamaño\ del\ vocabulario}\right)$$

Clasificación

Finalmente, para la última parte de la práctica, se debe crear un programa final que tome como entrada las estimaciones de probabilidad de cada palabra y un corpus, del tipo corpustodo.txt, y devolviendo un nuevo fichero clasificación.txt que contenga el tipo de mensaje con la etiqueta según su clasificación y el propio mensaje.

Para esta parte final, se cogió el corpus y se hizo la limpieza de la primera parte, es decir, quitando las URLs, y las palabras que precedían a los símbolos @ y #. El siguiente paso fue calcular las probabilidades generales de cada corpus, en nuestro caso P(A), P(D) y P(I). Para ello se hizo uso de la siguiente fórmula.

$$\log\left(\frac{número\ de\ tweets\ del\ corpus}{número\ de\ tweets\ total}\right)$$

Seguidamente, se pasó a ir consultando cada uno de los tweets que aparecían en el fichero a clasificar. De forma que, por cada palabra de dicho tweet se buscaba su probabilidad dentro del fichero de aprendizaje correspondiente al primer corpus, se sumaban todas ellas y se le sumaba la probabilidad general del corpus sobre el que se está trabajando. Este proceso se realiza con cada uno de los ficheros de aprendizaje asociados a los diferentes corpus.

Finalmente, para determinar en cual de ellos se clasifica el tweet se buscó el valor máximo entre dichas sumas de probabilidades. Por ejemplo, con un tweet con n palabras y para el corpus de acción la suma quedaría de la siguiente forma:

$$CorpusA\ \rightarrow \log(palabra1 \mid A) + \cdots + \log(palabraN \mid A) + \log(\mathrm{P(A)})$$

Aparte de trabajar en el programa de clasificación decidimos modificar un aspecto de los programas anteriores buscando una mejor precisión en la clasificación. La principal modificación fue observar las palabras más comunes en el lenguaje utilizado (como pueden ser “the” o “for”) e ignorarlas pues podían hacer que la calidad de clasificación disminuyese, como era el caso.

Por último, una vez realizada la clasificación, se calculó la probabilidad de acierto en cada uno de los corpus y la probabilidad total de acierto, sabiendo así si nuestro clasificador era bueno o malo. En nuestro caso, las probabilidades que salieron fueron:

- Información: 91.04 %
- Diálogo: 81.46 %
- Acción: 96.93 %
- Total: 92.13 %

A pesar de que esta probabilidad total no fue la más alta obtenida pues se llegó a tener un caso con un 94% modificando la lista de palabras más usuales del lenguaje nos decantamos por estos resultados debido a que para cada corpus estaban más equilibrados pues con el 94% de acierto los mensajes de información y acción se clasificaban mejor, pero los de diálogo mucho peor.

Estructura usada

El programa final cuenta con los siguientes ficheros:

- Corpustodo → fichero que contiene todos los mensajes después de la limpieza inicial.
- Vocabulario → fichero .txt que contiene todas las palabras encontradas en el corpus.
- CorpusI, CorpusA, CorpusD → distintos corpus que se utilizan para clasificar los mensajes.
- AprendizajeI, AprendizajeA, AprendizajeD → ficheros que contienen cada una de las palabras encontradas, con su frecuencia y su probabilidad.
- Clasificacion.txt → fichero que contiene cada uno de los mensajes clasificados por su tipo, según el programa.
- Aprendizaje.java → programa que genera un fichero, recibiendo dos como entrada (vocabulario y texto a clasificar), con las probabilidades de cada palabra del vocabulario.
- Clasificacion.java → programa que, recibiendo un vocabulario y unos ficheros de aprendizaje, clasifica en otro fichero el aprendizaje, es decir, comenta el tipo del mensaje.
- Vocabulario.java → programa que, recibiendo un fichero de entrada, separa el vocabulario de este.

Utilización de librerías

Como ya se mencionó anteriormente la implementación de todos los programas fue realizada en Java haciendo uso de Eclipse como IDE para desarrollarlos.

No se hizo uso de ninguna librería externa a Java, pero si cabría destacar el uso de algunas clases. La primera sería la clase TreeSet que nos ha permitido a la hora de crear el vocabulario tener una colección que controle directamente al añadir un nuevo elemento si ya existe un elemento igual, no añadirlo, además de que hace que se ordenen de forma automática al introducirse, es decir, se obtiene una colección de elementos sin duplica y ordenados mediante un criterio, en nuestro caso, orden alfabético.

Otra de las clases utilizadas ha sido ArrayList para poder crear arrays dinámicos, es decir, no tener que especificar un tamaño fijo desde un inicio para realizar, por ejemplo, el almacenamiento de palabras con las que trabajar a posteriori.

Por último, la lectura y escritura de ficheros se realizó mediante BufferedReader y BufferedWriter respectivamente, dos clases muy completas en cuanto a funcionalidades para realizar este tipo de tareas.

Porcentaje de trabajo

Para concluir el informe, se presenta a continuación el porcentaje de trabajo de cada uno de los integrantes del grupo en las distintas tareas a realizar.

	Programa	Informe
Cristina Garrido Amador	40%	60%
Jesús Ramos Álvarez	60%	40%

Como último añadido se incluye el enlace al GitHub dónde se pueden encontrar todos los programas y ficheros generados para esta práctica.

https://github.com/alu0100904932/IA---Lenguaje.git