Procesamiento de lenguaje natural

El lenguaje de programación empleado en la realización de esta práctica ha sido Kotlin, un lenguaje de tipado estático plenamente interoperable con Java que compila al bytecode de la JVM.

Procedimiento de trabajo

1 — División del fichero inicial

- **Entrada:** Instancia inicial: fichero CSV
- **Salida:** Dos corpus sin procesar: `unprocessed_corpusT` y `unprocessed_corpusNT`

El archivo con el que nos encontramos al inicio del proyecto es un CSV que contiene dos columnas, la primera siendo el contenido del tweet y la segunda la clasificación de dicho tweet, entre `T` *(troll)* o `NT` *(not_troll)*.

Esta fase de procedimiento es sencilla. Consiste en dividir el CSV inicial en dos corpus sin procesar: uno para el corpus de **trolls**, y otro para el corpus de **no trolls**.

2 — Procesamiento del corpus y generación del vocabulario

Procesamiento del corpus

- **Entrada:** `corpusT` y `corpusNT` sin procesar
- **Salida:** `corpusT` y `corpusNT` procesados, más `corpustodo` y el vocabulario conjunto de ambos corpus

---

Esta fase limpia los corpus sin procesar, realizando una determinada cantidad de modificaciones, que se mencionan a continuación.

- **Eliminacion de URLs** Las URLs de los tweets son suprimidas por completo. Las URLs pueden provenir de imágenes que acompañan al tweet o los propios enlaces que cuelgan los usuarios.
- **Eliminación de códigos ASCII** Los códigos ASCII son suprimidos. Ejemplos: `&#32;`, `&oacute;`. Normalmente son caracteres especiales que no aportan demasiada información.
- **Eliminación de menciones de usuarios** Ejemplo: `@usernamehere`. No aportan información. Deben ir precedidos y sucedidos de espacios.
- **Eliminación de símbolos perdidos** Los símbolos sueltos en medio de los tweets son eliminados. Un símbolo suelto se considera aquel símbolo que no es una letra o número y está rodeado de espacios (al menos uno) por ambos lados.

Una vez el corpus ha sido procesado con las modificaciones anteriores, se genera `corpusT` y `corpusNT`, mientras que `corpustodo` se genera a partir de la concatenación de `corpusT` y `corpusNT`.

Generación del vocabulario: tokenización

El vocabulario se genera a partir de la división de los tweets de `corpusT` y `corpusNT` (o en su defecto, `corpustodo`) en palabras (o *tokens*). Esta división es un proceso en sí mismo, sobre el cual también se realizan modificaciones:

- **División en palabras** Un tweet es una secuencia de palabras. Una palabra es considerada una secuencia de símbolos, separados por espacios o símbolos de puntuación (como puntos, comas, dos puntos, puntos y comas, símbolos de interrogación y símbolos de exclamación). Dichos caracteres son seleccionados para la división e inmediatamente descartados.
- **Eliminación de palabras conformadas por solo números** No aportan información útil.
- **Eliminación de palabras conformadas por solo símbolos** No brindan especial información, pero de ser tratadas correctamente podrían ser de utilidad.
- **Eliminación de símbolos alrededor de palabras** Dificultan la comparación entre las mismas palabras. Ejemplo: `'hey'` → `hey`.
- **Transformación a minúscula de palabras con mayúscula inicial** Ayuda a la comparación entre palabras. Ejemplo: Ejemplo: `Maria` → `maria`. Solo se aplica si la primera letra es mayúscula y el resto minúscula. En caso contrario, se devuelve la misma palabra (sin modificar). Se hace puesto que las palabras iniciales de las oraciones comienzan con mayúscula inicial, pero son la misma palabra aunque luego aparezcan todas en minúscula (en el caso de que no comiencen una frase). Por ejemplo, no tiene sentido clasificar `Car` y `car` como palabras distintas.

Finalmente, una vez se han generado todos los tokens del fichero, se suprimen las duplicidades y se genera el vocabulario de ambos corpus.

3 — Aprendizaje

- **Entrada:** `corpusT` y `corpusNT` procesados (+ vocabulario, en caso de que solo se provea un corpus)
- **Salida:** `aprendizajeT` y `aprendizajeNT`

Una vez se dispone de ambos corpus y el vocabulario, pasamos a calcular las probabilidades de que una palabra del vocabulario aparezca en alguno de los corpus.

Procedimiento por corpus

- Se tokeniza el corpus (mismo procedimiento que en la generación del vocabulario descrito en la sección anterior) pero sin que se supriman los duplicados. La tokenización es la lista de palabras del corpus.
- Se cuenta la cantidad de veces que aparece cada palabra del vocabulario en la tokenización del corpus y se guarda en un mapa `Palabra → Apariciones`. A dicho mapa se le añade la palabra especial para el tratamiento de palabras desconocidas: `<UNK>`, cuya cantidad de apariciones es 0. Si una palabra del vocabulario no figura en el corpus, se le da la frecuencia de `<UNK>`.
- Se calcula la probabilidad de que dicha palabra `x` aparezca, con la siguiente fórmula: $P(x) = \frac{c(x)+1}{N+|V|+1}$, donde $c(x)$ es la cantidad de veces que aparece esa palabra, N es la cantidad de tokens del corpus, $|V|$ es el tamaño del vocabulario, el 1 del numerador es por el suavizado laplaciano, y el 1 del denominador es por la inclusión de `<UNK>` en el vocabulario.
- Finalmente, se guarda $P(x)$ como $log(P(x))$ para evitar problemas de underflow a la hora de calcular la probabilidad más adelante, en la fase de clasificación.

Una vez se ha aplicado el procedimiento en ambos corpus, se almacena la palabra, la frecuencia y la probabilidad logarítmica en el respectivo fichero de aprendizaje.

4 — Clasificación

- **Entrada:** un corpus para clasificar, `aprendizajeT` y `aprendizajeNT`
- **Salida:** un fichero que indica a que clase pertenece cada uno de los tweets del corpus para clasificar

---

Procedimiento

- El corpus es procesado y separado en tweets.
- Cada uno de los tweets es tokenizado individualmente.
- Para cada una de las palabras del tweet se obtiene la probabilidad de que aparezca en `corpusT` y en `corpusNT` de acuerdo con los ficheros de aprendizaje (son dos variables distintas: `probTweetT` y `probTweetNT`). Luego se suman todas esas probabilidades entre sí (recordemos que son probabilidades logarítmicas) y finalmente se le suma la probabilidad de clase (probabilidad de troll o no troll). Básicamente:

  `probTweetX = Σ (probability of every tweet's word to apear in corpus X) + (probX)`

  donde `probTweetX` es la probabilidad del tweet de ser troll o no troll.
- Luego se comparan ambas probabilidades (la de ser troll o no troll) y se clasifica el tweet según cual sea mayor. Dicho resultado es escrito a un fichero en formato `T` (es troll) o `nT` (no es troll).

Comprobación de la clasificación

Probando corpus aleatorios el porcentaje de clasificación ronda el 88 por ciento.

Sobre el código

El código está pensado para trabajar con ficheros ya existentes en `src/main/resources`, pero las rutas se pueden cambiar desde `src/main/kotlin/App.kt`.

Descripción de archivos

Archivo de referencia: `ficheros.zip`

- `aprendizajeT.txt`: fichero de aprendizaje de los trolls
- `aprendizajeNT.txt`: fichero de aprendizaje de los no trolls
- `clasificacion.txt`: resultado de clasificar `corpustodo`
- `clasificaciontext.txt`: resultado de clasificar `corpustest`
- `corpusNT.txt`: corpus de no trolls
- `corpusT`.txt: corpus de trolls
- `corpustest`.txt: corpus del profesor
- `corpustodo`.txt: concatenación de `corpusT` y `corpusNT`
- `vocabulary.txt`: vocabulario de `corpustodo`