Natural language processing (NLP)

Procesamiento del lenguaje natural (PLN)

Primero, por **lenguaje natural vamos a entender el lenguaje humano (lenguas del mundo).**

¿Qué es? Es un área de la computación, IA y lingüistica que estudia las interacciones entre los computadores y el lenguaje humano.

¿Cuál es objetivo del PLN? Diseñar algoritmos que le permitan a los computadores "entender" el lenguaje natural para realizar alguna tarea. Ejemplos de tareas con diferentes niveles de dificultad son:

- Dificultad baja: corrección ortográfica, búsqueda de palabras claves, encontrar sinónimos,
- Dificultad media: analizar información de documentos o sitios web
- Dificultad alta: traducción mecánica, análisis semántico, ¿esto, este, el, ella...? encontrar a qué o quién se hace referencia, responder preguntas

¿Cómo representar las palabras como entrada para los modelos? Se utiliza una representación vectorial: la idea es que si V es un vocabulario con el que se está trabajando, $V = \{árbol, planta, bosque, \ldots\}$ decimos que una función $f : V \to \mathbb{R}^n$ induce una representación vectorial de dimensión n para dicho vocabulario, si le asigna un vector de $\mathbb{R}^n$ a cada palabra de V.

La construcción de funciones f que tengan buenas propiedades es objeto de estudio, pues el desempeño de una gran cantidad de algoritmos de PLN dependen directamente de la calidad de las mismas. Dependiendo el contexto en que se esté trabajando,uni puede desear que f satisfaga ciertas propiedades, por ejemplo que:

- palabras con significados similares tengas asociados vectores 'similares',
- capturar la relación entre dos palabras como por ejemplo *cantar* y *cantando,*
- Un n no muy grande $(n \ll |V|)$ para que la dimensionalidad de las entradas no sea un problema ó alta dimensionalidad pero buscando que los vectores asociados a las palabras tengan casi todas sus entradas en cero.

Nosotros los haremos usando TF- IDF (term frequency - inverse document frequency). Pero primero necesitamos introducir el siguiente concepto:

Palabras vacías (stop words). Son palabras que se filtran antes o después del procesamiento de datos en lenguaje natural. Usualmente el término hace referencia a las palabras más comunes en un idioma.

Volviendo a nuestra forma de vectorizar:

1. **TF:**

   ```
   documento 1 : el cielo es azúl
   documento 2 : el sol es brillante
   ```

   Ignoramos las palabras vacías y esto nos da un vector para cada documento:

```
             azúl brillante cielo sol
documento 1     1      0       1    0
documento 2     0      1       0    1
```

2. **IDF:**

$$\log\left(\frac{1+\text{cantidad de documentos}}{1+\text{número de documentos donde aparece el término}}\right)+1$$

Por ejemplo IDF de cielo:

$$\log\left(\frac{1+2}{1+1}\right)+1=1.40$$

3. **TF - IDF:**

$$\text{TF-IDF}=\text{TF}\times\text{IDF},$$

donde cada entrada de la matriz se normaliza con la norma ℓ_2

```
              azúl      brillante      cielo      sol
documento 1   0.7071         0         0.7071       0
documento 2      0        0.7071          0      0.7071
```

Multinomial Naive Bayes Classifier for text analysis
Clasificador Naive Bayes Multinomial para análisis de texto

Supongamos que el problema es el siguiente: clasificar documentos con una representación vectorial.

Este modelo es un supervisado y por lo tanto necesitamos conjunto de entrenamiento y conjunto de prueba.

Supongamos que tenemos $|V|$ palabras conformando el vocabulario.

El modelo se basa en la aplicación de la regla de Bayes para predecir la probabilidad condicional de que un documento d_j pertenezca a una clase c_i a partir de la probabilidad de los documentos dada la clase y la probabilidad a priori de la clase en el conjunto de entrenamiento:

$$P(c_i|d_j)=\frac{P(c_i)\cdot P(d_j|c_i)}{P(d_j)}$$

$P(d_j)$ es la probabilidad de que al menos una palabra del vocabulario esté en el documento, es decir, es 1.

Ahora, la probabilidad de un documento dada la clase suele asumirse como la probabilidad conjunta de las palabras que aparecen en dichos documentos dada la clase y se calculan como

$$P(d_j|c_i) = \prod_{t=1}^{|V|} P(w_t|c_i).$$

Adicionalmente, el modelo considera la frecuencia de aparición de cada palabra en los documentos, x_t, en vez de una ocurrencia binaria

$$P(d_j|c_i) = \prod_{t=1}^{|V|} P(w_t|c_i)^{x_t}.$$