

# Natural Language Processing Vectors Space Model and Information Retrieval

Felipe Bravo-Marquez

July 12, 2019

# Motivation

- How does a search engine such as Duckduckgo or Google retrieve relevant documents from a given query?
- How can a company process the claims left by its users on its Web portals?

The following areas of knowledge study these problems:

- *Information Retrieval*: science of searching for information in a documents.
- *Text Mining*: Extraction of knowledge from text.

Both of them are closely related to NLP! (the borders between these fields are unclear)

# Tokens and Types

Tokenization: The task of splitting a sentence or document into *tokens*. Additional transformations can be employed such as the removal of special characters (e.g., punctuation), lowercasing, etc. [Manning et al., 2008].

## Example

Input: I like human languages and programming languages.

Tokens: [I] [like] [human] [languages] [and] [programming] [languages]

A *type* is a class of *token* containing a single sequence of characters. They are obtained by identifying unique tokens within the document.

## Types

Tipos: [I] [like] [human] [languages] [and] [programming]

*The token languages was repeated in the sentece*

# Vocabulary Extraction

A *term* is a normalized *type*. The vocabulary  $V$ , is the set of terms (normalized unique tokens) within a collection of documents or corpus  $D$ .

## Stopwords removal

- In order to reduce the size of the vocabulary and eliminate terms that do not provide much information, terms that occur with high frequency are in the corpus are eliminated.
- These terms are called *stopwords* and include articles, pronouns, prepositions and conjunctions.  
Example: [a, an, and, any, has, do, don't, did, the, on].<sup>1</sup>

The removal of stopwords can be inconvenient in many NLP tasks!!

Example: I don't like pizza => pizza ( "I", "don't", and "like" were removed)

---

<sup>1</sup>Related concepts: function words, closed-class words.

# Stemming

A process in which terms are transformed to their root in order to reduce the size of the vocabulary. It is carried out on the basis of a set of word reduction rules.

Example: Porter's Algorithm.

(F)	Rule	Example
	SSES → SS	caresses → caress
	IES → I	ponies → poni
	SS → SS	caress → caress
	S →	cats → cat

Example:  $d$  = I like human languages and programming languages  $\Rightarrow$  I like human languag and program languag<sup>2</sup>

The vocabulary of document  $d$  after removing stopwords and performing stemming:

termId	value
t1	human
t2	languag
t3	program

---

<sup>2</sup>[http://9ol.es/porter\\_js\\_demo.html](http://9ol.es/porter_js_demo.html)

# Lemmatization

- Another strategy to transform words to their root. It performs a morphological analysis using reference dictionaries (lookup tables) to create equivalence classes between *types*.
- For example for the token *saw*, a stemming rule could return the term *s*, while through lemmatization a dictionary would give us *see*.

## Zipf's law [1]

- The Zipf's law, proposed by *George Kingsley Zipf* in [Zipf, 1935], is an empirical law about the frequency of terms within a collection of documents (corpus).
- It states that the frequency  $f$  of a term in a corpus is inversely proportional to its  $r$  ranking in a sorted frequency table:

$$f = \frac{cf}{r^\beta} \quad (1)$$

- Where  $cf$  is a constant dependent on the collection and  $\beta > 0$  is a decay factor.
- If  $\beta = 1$ , then  $f$  follows exactly Zipf's law, otherwise it follows a Zipf-like distribution.
- The law relates to the principle of minimum effort. We often use a few words to write ideas.
- The Zipf law is a type of power law distribution (long tail distributions)

## Zipf's law [2]

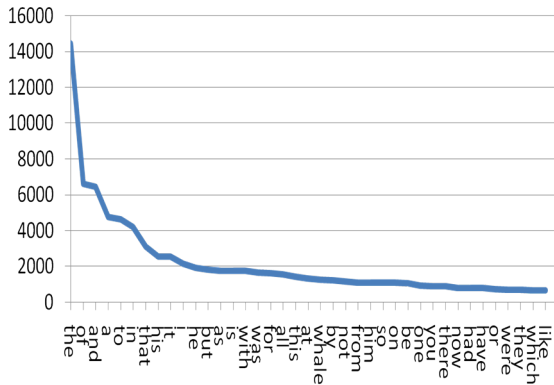


Figure: Zipf's law

- If we plot a  $\log - \log$  graph, we obtain a straight line with slope  $-\beta^{-1}$ .
- Most frequent words can be used to build a *stopwords* list.



# Posting List and Inverted Index

Let  $D$  be a collection of documents and  $V$  the vocabulary of all terms extracted from the collection:

- The posting list of a term is the list of all documents where the term appears at least once.
- An inverted index is a dictionary-type data structure mapping terms  $t_i \in V$  into their corresponding posting lists.

$$\langle term \rangle \rightarrow \langle docId \rangle^*$$

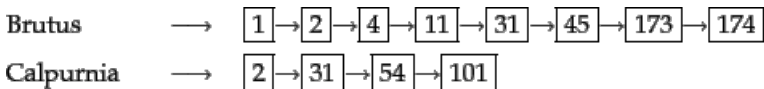


Figure: Inverted Index

# Web Search Engines [1]

Un motor de búsqueda es un sistema de recuperación de información diseñado para la búsqueda de información en la Web [Gutiérrez et al., 2008]. Sus componentes básicos son:

- Crawler: Un robot que navega la Web según una estrategia definida. Generalmente comienza navegando por un conjunto de páginas semilla (seeds) y continua navegando por sus hipervínculos.
- Indexador: Encargado de mantener un índice invertido con el contenido de las páginas recorridas por el Crawler.
- Máquina de consultas: Encargado de procesar las consultas y buscar en el índice los documentos con mayor similitud a ella.
- Función de ranking: Es la función que tiene la máquina de consulta para rankear los documentos indexados en la colección por relevancia para una consulta.
- Interfaz: Interactúa con el usuario, recibe la consulta como entrada y retorna los documentos rankeados por similitud.

# Web Search Engines [2]

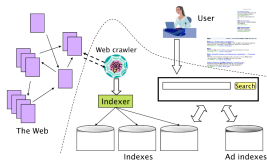


Figure: Diagrama Motor de Búsqueda [Gutiérrez et al., 2008].

# Vector Space Models

- Para poder rankear las consultas, o medir la similitud entre dos documentos necesitamos una métrica de similitud.
- Representamos los documentos como vectores de términos, donde cada término es una dimensión.
- A este tipos de modelos se le llame *Bag of Words*. Perdemos el orden de las palabras.
- El valor de cada dimensión, es un peso que representa la relevancia del término  $t_i$  en el documento  $d$ .

$$d_j \rightarrow \vec{d_j} = (w(t_1, d_j), \dots, w(t_{|V|}, d_j)) \quad (2)$$

- ¿Cómo podemos modelar el aporte de información de un término en un documento?

# Term Frequency - Inverted Document Frequency [1]

- Se define  $Tf_{i,j}$ , como la frecuencia del término  $t_i$  en el documento  $d_j$ .
- Un término que aparece 10 veces debiese aportar mayor información que uno aparece una vez.
- ¿Qué pasa cuando tenemos documentos muchos más largos que otros?
- Podemos normalizar por la frecuencia máxima de término en el documento.

$$Tf_{i,j} = \frac{f_{i,j}}{\max f_{i,j}}$$

- ¿Un término que aparece en muy pocos documentos aporta más o menos información que uno que aparece varias veces?
- Por ejemplo, el documento *El señor alcalde de Malloco*. El término *Malloco* aparece en menos documentos que *alcalde*, por lo que debiese ser más descriptivo.

## Term Frequency - Inverted Document Frequency [2]

- Sea  $N$  el número de documentos en la colección y  $n_i$  el número de documentos donde aparece el término  $t_i$ , se define el *idf* del término  $t_i$  como:

$$idf_{t_i} = \log_{10}\left(\frac{N}{n_i}\right)$$

- Un término que aparece en todos los documentos tendría  $idf = 0$  y uno que aparece en el 10% de la colección tendría  $idf = 1$ .
- El modelo de score  $Tf - idf$  combina ambos modelos, quedando el peso  $w$  de un término sobre un documento como:

$$w(t_i, d_j) = Tf_i \times \log_{10}\left(\frac{N}{n_i}\right)$$

- Las consultas a un motor de búsqueda también pueden modelarse como vectores, pero las consultas tienen en promedio entre 2 y 3 términos. Para evitar tener tantas dimensiones nulas, se usa un factor de suavizamiento en el vector:

$$w(t_i, d_j) = (0.5 + 0.5 \times Tf_{i,j}) \log_{10}\left(\frac{N}{n_i}\right)$$

## Similarity between Vectors

- Una vez representados, los documentos y consultas como vectores, podemos medir su similitud.
- Una alternativa sería usar la distancia euclidiana, pero la variabilidad de largo entre documentos afectaría a la métrica.
- Lo más usado es usar el coseno del ángulo entre los vectores como medida de similitud.
- Si los documentos son iguales, el ángulo vale 0 y el coseno 1. En cambio si son ortogonales el coseno vale 0.
- Los vectores, deben ser normalizados por su norma euclidiana  $\|d\|_2$ , la similitud se calcula de la siguiente manera:

$$\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1| \times |d_2|} = \frac{\sum_{i=1}^{|V|} (w(t_i, d_1) \times w(t_i, d_2))}{\sqrt{\sum_{i=1}^{|V|} w(t_i, d_1)^2} \times \sqrt{\sum_{i=1}^{|V|} w(t_i, d_2)^2}}$$

- Erróneamente se llama *distancia coseno*, realmente es una medida de similitud.

## Cosine Similarity

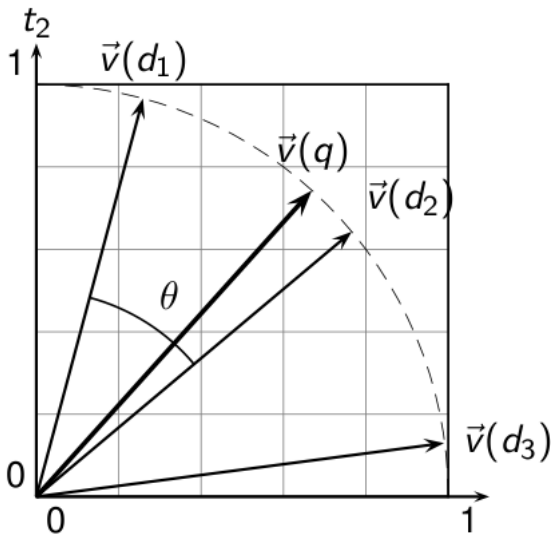


Figure: Cosine Similarity.



## Exercise

- Supongamos que tenemos 3 documentos, los cuales se forman a partir de las siguientes secuencias de términos:  
 $d_1 \rightarrow t_4 t_3 t_1 t_4$   
 $d_2 \rightarrow t_5 t_4 t_2 t_3 t_5$   
 $d_3 \rightarrow t_2 t_1 t_4 t_4$
- Construya una matriz término-documento de dimensión  $5 \times 3$  usando los pesos *Tf – idf* simples (sin normalización).
- Le recomendamos construir primero una lista con la cantidad de documentos en los que aparece cada término (para el *idf*)
- Calcule luego el *idf* de cada término.
- Llene las celdas con los valores *Tf – idf*
- ¿ A qué documento está más cercano  $d_1$  ?

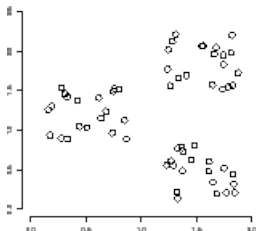
# Result

Table: Tf-idf Matrix

	d1	d2	d3
t1	0.176	0.000	0.176
t2	0.000	0.176	0.176
t3	0.176	0.176	0.000
t4	0.000	0.000	0.000
t5	0.000	0.954	0.000

## Clustering de Documentos [1]

- ¿Qué pasa si queremos agrupar los documentos de contenidos similares?
- Agrupamos los documentos en conjuntos, donde todos los elementos sean similares entre sí.
- A cada conjunto se le llama *cluster*.
- El problema de clusterizar, se basa en identificar grupos que maximicen la similitud interna dentro de un cluster y minimicen la similitud entre documentos pertenecientes a distintos clusters.



**Figure:** Conjunto de documentos donde se identifica claramente cada cluster

## Clustering de Documentos [2]

- Permite identificar grupos de opiniones similares, o reducir el espacio de búsqueda para una consulta en un buscador.
- K-medias es un algoritmo simple de clustering que requiere la cantidad  $k$  de clusters a construir como parámetro.
  1. Primero, se identifican aleatoriamente  $k$  elementos. Los valores de los atributos de éstos elementos se copian en nuevos elementos llamados centroides de la misma dimensión que éstos. Cada centroide representará un cluster.
  2. Luego se calcula la distancia de todos los  $n$  elementos a los  $k$  centroides y se asigna cada elemento al cluster del centroide más cercano.
  3. Luego se recalcula el valor de los centroides promediando el valor de los atributos de todos los elementos pertenecientes al cluster.
  4. Repite el proceso de calcular las distancias, agrupar los más cercanos y recalcular los centroides hasta que éstos dejen de cambiar.

## K-medias

```
K-MEANS( $\{\vec{x}_1, \dots, \vec{x}_N\}, K$ )
1   $(\vec{s}_1, \vec{s}_2, \dots, \vec{s}_K) \leftarrow \text{SELECTRANDOMSEEDS}(\{\vec{x}_1, \dots, \vec{x}_N\}, K)$ 
2  for  $k \leftarrow 1$  to  $K$ 
3  do  $\vec{\mu}_k \leftarrow \vec{s}_k$ 
4  while stopping criterion has not been met
5  do for  $k \leftarrow 1$  to  $K$ 
6      do  $\omega_k \leftarrow \{\}$ 
7      for  $n \leftarrow 1$  to  $N$ 
8          do  $j \leftarrow \arg \min_j |\vec{\mu}_j - \vec{x}_n|$ 
9               $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$  (reassignment of vectors)
10     for  $k \leftarrow 1$  to  $K$ 
11         do  $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$  (recomputation of centroids)
12 return  $\{\vec{\mu}_1, \dots, \vec{\mu}_K\}$ 
```

Figure: Algoritmo K-medias

# References I



Gutiérrez, C., Navarro, G., Baeza-Yates, R., Hurtado, C., Arenas, M., Marín, M., Piquer, J. M., Rodríguez, M., del Solar, J. R., and Velasco, J. (2008).  
*Cómo funciona la Web*.  
Autoeditada.



Manning, C. D., Raghavan, P., and Schütze, H. (2008).  
*Introduction to Information Retrieval*.  
Cambridge University Press, New York, NY, USA.



Zipf, G. K. (1935).  
*The Psychobiology of Language*.  
Houghton-Mifflin, New York, NY, USA.