

# Natural Language Processing

## Vector Space Model and Information Retrieval

Felipe Bravo-Marquez

March 22, 2020

# 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?

These problems are studied in the following fields:

- *Information Retrieval*: science of searching for information in document collections.
- *Text Mining*: automatic 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 pieces called *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]

## Types

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

Types for the previous sentence: [I] [like] [human] [languages] [and] [programming]

The token *languages* was repeated in the sentence.

# Vocabulary Extraction

- A *term* is a normalized *type*.
- Normalization is the process of creating equivalence classes of different *types*. This will become clear in the following slides.
- 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 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 term normalization process in which terms are transformed to their root in order to reduce the size of the vocabulary. It is carried by applying word reduction rules.

Example: Porter's Algorithm.

(F)

**Rule**  
SSES → SS  
IES → I  
SS → SS  
S →

**Example**

caresses → caress  
ponies → poni  
caress → caress  
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 term normalization strategy.
- It also transform words into their roots.
- It performs a morphological analysis using reference dictionaries (lookup tables) to create equivalence classes between *types*.
- For example, for the token *studies*, a stemming rule would return the term *studi*, while through lemmatization we would get the term *study*<sup>3</sup>.

---

<sup>3</sup><https://blog.bitext.com/>

## 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:

$$f = \frac{cf}{r^\beta}$$

(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]

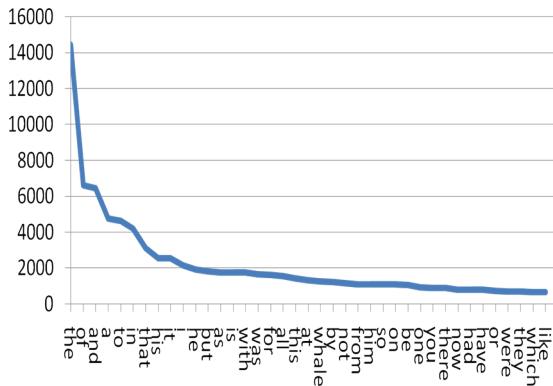


Figure: Zipf's law

- If we plot a *log-log* graph, we obtain a straight line with slope  $-\beta$ .
- Listing the most frequent words of a corpus can be used to build a *stopwords* list.



# Posting Lists and the 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. Documents are identified by their ids.
- An inverted index is a dictionary-type data structure mapping terms  $t_i \in V$  into their corresponding posting lists.

$$\langle \text{term} \rangle \rightarrow \langle \text{docId} \rangle^*$$

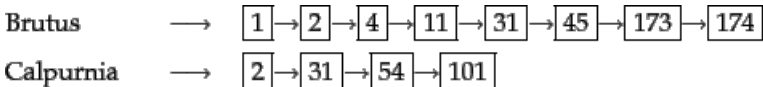


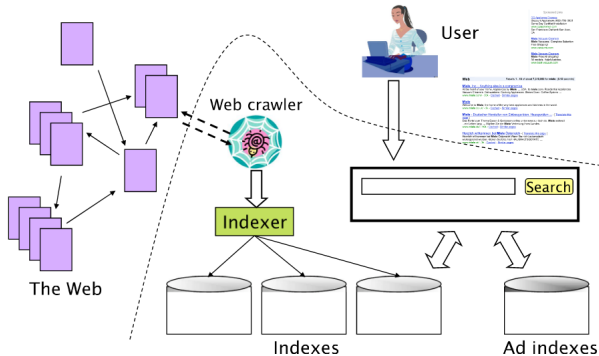
Figure: Inverted Index

# Web Search Engines [1]

A search engine is an information retrieval system designed for searching information on the Web (solving information needs) [Manning et al., 2008]. Its basic components are:

- **Crawler**: a robot that navigates the Web according to a defined strategy. It usually starts by browsing a set of seed websites and continues to browse their hyperlinks.
- **Indexer**: in charge of maintaining an inverted index with the content of the pages traversed by the Crawler.
- **Query processor**: in charge of processing user queries and searching the index for the documents most relevant to a query.
- **Ranking function**: the function used by the query processor to rank documents indexed in the collection by relevance according to a query.
- **User interface**: receives the query as input and returns the documents ranked by relevancy.

# Web Search Engines [2]



**Figure:** The various components of a web search engine [Manning et al., 2008].

# The Vector Space Model

- In order to rank queries, or measure the similarity between two documents we need a **similarity metric**.
- **Documents** can be **represented as vectors of terms**, where **each term** is a **vector dimension** [Salton et al., 1975].
- Documents with different words and lengths will reside in the same vector space.
- These types of representations are called **Bag of Words**.
- In **bag-of-words-representations** the **order of words** and the **linguistic structure** of a sentence is **lost**.
- The **value of each dimension** is a **weight that represents the relevance** of the **term  $t_i$**  in the **document  $d$** .

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

- How can we model how informative is a term to a document?

# Term Frequency - Inverted Document Frequency [1]

- Let  $tf_{i,j}$  be the frequency of term  $t_i$  in document  $d_j$ .
- A term that occurs 10 times should provide more information than one that occurs once.
- What happens when we have documents that are much longer than the others?
- We can normalize by the maximum term frequency in the document.

$$ntf_{i,j} = \frac{tf_{i,j}}{\max_i(tf_{i,j})}$$

- Does a term that occurs in very few documents provide more or less information than one that occurs several times?
- For example, the document *The respected major of Pelotillehue*. The term *Pelotillehue* occurs in fewer documents than the term *major*, so it should be more descriptive.

# Term Frequency - Inverted Document Frequency [2]

- Let  $N$  be the number of documents in the collection and  $n_i$  the number of documents containing term  $t_i$ , we define  $idf$  of  $t_i$  as follows:

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

- A term that appears in all documents would have  $idf = 0$  and one that appears in 10% of the documents would have  $idf = 1$ .
- The  $tf-idf$  scoring model combines  $tf$  and  $idf$  scores, resulting in the following weights  $w$  for a term in a document:

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

- Search engine queries can also be modeled as vectors. However, queries have between 2 and 3 terms in average. To avoid having too many null dimensions, query vectors can be smoothed as follows:

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

## Similarity between Vectors

- Representing queries and documents as vectors allows calculating their similarity.
- One approach would be using the euclidean distance.
- The common approach is to calculate the cosine of the angle between the two vectors.
- If both documents are the same, the angle would be 0 and its cosine would be 1. On the other hand, if they are orthogonal the cosine is 0.
- The cosine similarity is calculated as follows:

$$\cos(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| \times |\vec{d}_2|} = \frac{\sum_{i=1}^{|V|} (w(t_i, d_1) \times w(t_i, d_2))}{\sqrt{\sum_{i=1}^{|V|} w(t_i, d_1)^2} \times \sqrt{\sum_{i=1}^{|V|} w(t_i, d_2)^2}}$$

- This is wrongly called *cosine distance*. It is actually a similarity metric.
- Notice that cosine similarity normalizes the vectors by its euclidean norm  $||\vec{d}||_2$ .

## Cosine Similarity

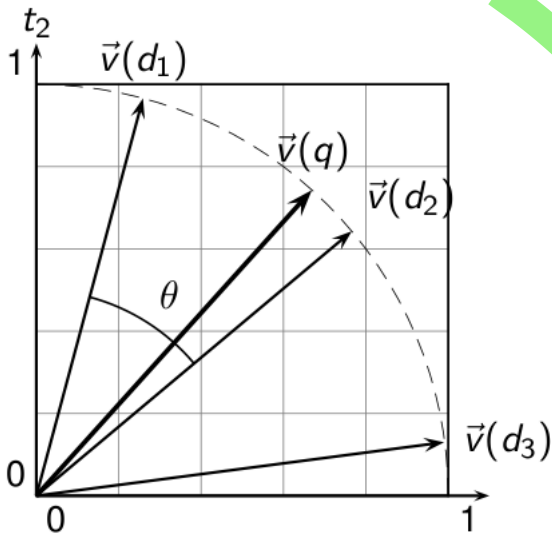


Figure: Cosine Similarity.



## Exercise

- Suppose we have 3 documents formed from the following sequences of terms:

$$d_1 \rightarrow t_4 t_3 t_1 t_4$$

$$d_2 \rightarrow t_5 t_4 t_2 t_3 t_5$$

$$d_3 \rightarrow t_2 t_1 t_4 t_4$$

- Build a term-document matrix of  $5 \times 3$  dimensions using simple *tf-idf* weights (without normalization).
- We recommend you first build a list with the number of documents in which each term appears (useful for calculating *idf* scores)
- Then calculate the *idf* scores of each term.
- Fill up the cells of the matrix the *tf-idf* values.
- Which is the closest document to  $d_1$ ?

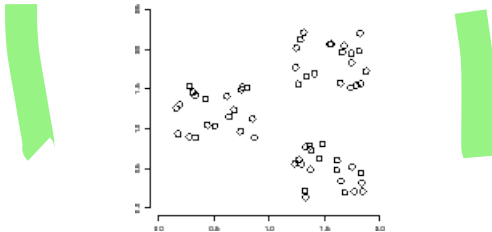
# Result

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

# Document Clustering [1]

- How can we group documents that are similar with each other?
- **Clustering** is the **process of grouping documents** that are **similar** with each other.
- **Each group** of documents is called a **cluster**.
- In clustering we try to identify **groups of documents** in which the **similarity between documents in the same cluster is maximized** and the **similarity of documents in different clusters is minimized**.



**Figure:** Set of documents where the clusters can be clearly identified.

## Document Clustering [2]

- Document clustering allows identifying topics in a corpus and reducing the search space in a search engine i.e., the inverted index is organized according to the clusters.
- K-means is a simple clustering algorithm that receives the number of clusters  $k$  as a parameter.
- The algorithm relies on the idea of *centroid*, which is the average vector of documents belonging to the same cluster.
- Let  $S$  be a set of 2-dimensional vectors  $\{3, 6\}$ ,  $\{1, 2\}$ ,  $\{5, 1\}$ , the centroid of  $S$  is  $\{(3 + 1 + 5)/3, (6 + 2 + 1)/3\} = \{3, 3\}$ .

# K-Means

1. We start with  $k$  random centroids.
2. We calculate the similarity between each document and each centroid.
3. We assign each document to its closest centroid forming a cluster.
4. The centroids are recalculated according to the documents assigned to them.
5. This process is repeated until convergence.

# K-means

```
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: K-means algorithm

# Conclusions and Additional Concepts

- Representing documents as vectors is essential for calculating similarities between document pairs.
- Bag of words vectors lack linguistic structure.
- Bag of words vectors are high-dimensional and sparse.
- Word n-grams can help capturing multi word-expressions (e.g., New York => new\_york)
- Modern information retrieval systems go beyond vector similarity (PageRank, Relevance Feedback, Query log mining, Google Knowledge Graph, Machine Learning).
- Information retrieval and text mining are less concerned with linguistic structure, and more interested in producing fast and scalable algorithms [Eisenstein, 2018].

# References I



Eisenstein, J. (2018).  
Natural language processing.  
Technical report, Georgia Tech.



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



Salton, G., Wong, A., and Yang, C.-S. (1975).  
A vector space model for automatic indexing.  
*Communications of the ACM*, 18(11):613–620.



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