

# Taller: Herramientas para la Minería de Textos (Parte 2)

Marcelo Errecalde<sup>1</sup>

<sup>1</sup>Universidad Nacional de San Luis, Argentina



II Congreso Internacional Innovación y  
Tecnología Informática en las Organizaciones



Cuenca, 8, 9 y 10 de Octubre de 2019

## Resumen

### 1 Representación de Documentos

- Atributos Estáticos
- Bolsa de Palabras
- Enfoques Distribucionales
- Aprendizaje de representaciones

### 2 Reducción de la dimensionalidad

- Indexado (análisis) de semántica latente

## ¿Qué características (features) usar para representar documentos?

## ¿Qué características (features) usar para representar documentos?

Depende del problema:

- ## ① ¿Categorización por tópico/tema?

## ¿Qué características (features) usar para representar documentos?

Depende del problema:

- ① ¿Categorización por tópico/tema?
  - ② ¿Categorización por autor?

## ¿Qué características (features) usar para representar documentos?

Depende del problema:

- 1 ¿Categorización por tópico/tema?
  - 2 ¿Categorización por autor?
  - 3 ¿Categorización del perfil del autor (sexo, nacionalidad, grupo etario)?

## ¿Qué características (features) usar para representar documentos?

Depende del problema:

- 1 ¿Categorización por tópico/tema?
  - 2 ¿Categorización por autor?
  - 3 ¿Categorización del perfil del autor (sexo, nacionalidad, grupo etario)?
  - 4 ¿Subjetividad/Objetividad, emociones?

## ¿Qué características (features) usar para representar documentos?

Depende del problema:

- 1 ¿Categorización por tópico/tema?
  - 2 ¿Categorización por autor?
  - 3 ¿Categorización del perfil del autor (sexo, nacionalidad, grupo etario)?
  - 4 ¿Subjetividad/Objetividad, emociones?
  - 5 ¿Identificación de pedófilos?

## Representación de Documentos

## Representaciones con características estáticas

Se eligen **antes** del procesamiento de los documentos.

# Representación de Documentos

## Representaciones con características estáticas

Se eligen **antes** del procesamiento de los documentos.

## Representaciones con características dinámicas

Surgen **como parte** del procesamiento de los documentos.

## Representaciones aprendidas

Extienden el uso del aprendizaje automático, también a la representación de los documentos

## Ejemplo: Textos arbitrarios - características estáticas

## Documentos

- 1 "pintaron el banco de la plaza"
  - 2 "te paso el programa, ejecútalo paso por paso"
  - 3 "sentado en el banco, miraba si el banco abría"

## Ejemplo: Textos arbitrarios - características estáticas

# Documentos

- 1 "pintaron el banco de la plaza"
  - 2 "te paso el programa, ejecútalo paso por paso"
  - 3 "sentado en el banco, miraba si el banco abría"

Número de palabras (NP), longitud de palabra más larga (LPL), longitud promedio de palabras (LPP), verbos en pasado (VP)

ID	NP	LPL	LPP	VP
d1	6	8	4	1
d2	8	9	4,5	0
d3	9	7	4	1



## Características estáticas: ejemplo en Wikipedia

Information Processing and Management 54 (2018) 1169–1181



Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: [www.elsevier.com/locate/infoproman](http://www.elsevier.com/locate/infoproman)



## Quality flaw prediction in Spanish Wikipedia: A case of study with verifiability flaws



Edgardo Ferretti<sup>a,b</sup>, Leticia Cagnina<sup>a,b,c</sup>, Viviana Paiz<sup>a</sup>, Sebastián Delle Donne<sup>a</sup>, Rodrigo Zacagnini<sup>a</sup>, Marcelo Erreca<sup>a,b</sup>

<sup>a</sup> Departamento de Informática, Universidad Nacional de San Luis (UNSL), Ejército de los Andes 950, San Luis, Argentina.

<sup>19</sup> Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (UNSL), Argentina.

<sup>c</sup> Consejo Nacional de Investigaciones, Científicas y Técnicas (CONICET), Argentina

## Atributos Estáticos

## Características estáticas: ejemplo en Wikipedia

**Table 1**  
Features that comprise the document model.

Feature	Description
<b>Content-based</b>	
Character count	Number of characters in the text (no spaces).
Word count	Number of words in the plain text.
Sentence count	Number of sentences in the plain text.
Word length	Average word length in characters.
Sentence length	Average sentence length in words.
Paragraph count	Number of paragraphs.
Paragraph length	Average paragraph length in sentences.
Longest word length	Length in characters of the longest word.
Longest sentence length	Number of words in the longest sentence.
Shortest sentence length	Number of words in the shortest sentence.
Long sentence rate	Percentage of long sentences. A long sentence is defined as containing at least 30 words.
Short sentence rate	Percentage of short sentences. A short sentence is defined as containing at most 15 words.
Longest subsection length	Length in words of the longest subsection.
Shortest subsection length	Length in words of the shortest subsection.
Subsections length	Total number of words in the article's subsections.
Average subsection length	Average number of words per subsection.
Longest subsubsection	Length in words of the longest subsubsection.
Shortest subsubsection	Length in words of the shortest subsubsection.
Subsubsections length	Total number of words in the article's subsubsections.
Average subsubsections	Average number of words per subsubsection.
<b>Structure-based</b>	
Section count	Number of sections.
Subsection count	Number of subsections.
Subsubsection count	Number of subsubsections.
Heading count	Number of sections, subsections and subsubsections.
Section nesting	Average number of subsections per section.
Subsection nesting	Average number of subsubsections per subsection.
Reference Sections Count	Number of reference sections, e.g. "References", "Footnotes", "Sources", "Bibliography".
Mandatory Sections Count	Number of mandatory sections, e.g. "See also".
Related page count	Number of related pages, e.g. "Further reading", "See also", etc.
Lead length	Number of words in the lead section (text before the first heading).
Lead rate	Percentage of words in the lead section.
Image count	Number of images.
Image rate	Ratio of image count to section count.
Link count	Every occurrence of a link (introduced with two open square brackets) in the unfiltered text.
Link rate	Percentage of links.
Table count	Number of tables.
Reference count	Number of all references using the <code>&lt;ref&gt;...&lt;/ref&gt;</code> syntax.
Reference section rate	Ratio of reference count to the accumulated section, subsection and subsubsection count.
Reference word rate	Ratio of reference count to word count.
Unique reference count	Number of unique references using the <code>&lt;ref&gt;...&lt;/ref&gt;</code> syntax.
Reference ratio	Ratio between the reference word rate of the article and the maximum reference word rate found in the dataset.
Templates-count	Number of (different) Wikipedia templates.

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

- **Objetos/instancias:** información de cada **texto**.

## Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

- **Objetos/instancias:** información de cada **texto**.
- **Atributos:** las distintas **palabras** que aparecen en los documentos.

## Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

- **Objetos/instancias:** información de cada **texto**.
- **Atributos:** las distintas **palabras** que aparecen en los documentos.
- **Valores de los atributos:** de tipo **numéricos** o booleanos.

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

- **Objetos/instancias:** información de cada **texto**.
- **Atributos:** las distintas **palabras** que aparecen en los documentos.
- **Valores de los atributos:** de tipo **numéricos** o **booleanos**.

**texto1.txt**

"pintaron el banco de la plaza"

**texto2.txt**

"te paso el programa, ejecútalo paso por paso"

**texto3.txt**

"sentado en el banco, miraba si el banco abría"

## Bolsa de Palabras

## Ejemplo: los documentos en Python ...

In [1]:

```
documentos = ["pintaron el banco de la plaza",
    "te paso el programa, ejecútalo paso por paso",
    "sentado en el banco, miraba si el banco abría"]
```

## Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

### Pesos *TF* (Frecuencia del término)

ID	abría	banco	de	ejecútalo	el	en	la	miraba	paso	pintaron	plaza	por	programa	sentado	si	te
d1	0	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
d2	0	0	0	1	1	0	0	0	3	0	0	1	1	0	0	1
d3	1	2	0	0	2	1	0	1	0	0	0	0	0	1	1	0

Bolsa de Palabras

## Peso = frecuencia del término (Python)

In [1]:

```
documentos = ["pintaron el banco de la plaza",
               "te paso el programa, ejecútalo paso por paso",
               "sentado en el banco, miraba si el banco abría"]
```

## Bolsa de Palabras

## Peso = frecuencia del término (Python)

In [1]:

```
documentos = ["pintaron el banco de la plaza",
              "te paso el programa, ejecútalo paso por paso",
              "sentado en el banco, miraba si el banco abría"]
```

In [2]:

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
vect.fit(documentos)
```

Bolsa de Palabras

## Peso = frecuencia del término (Python)

In [1]:

```
documentos = ["pintaron el banco de la plaza",
               "te paso el programa, ejecútalo paso por paso",
               "sentado en el banco, miraba si el banco abría"]
```

In [2]:

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
vect.fit(documentos)

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='',
                lowercase=True, max_df=1.0, max_features=None, min_df=,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(\\b\\w+\\b)', tokenizer=None,
                vocabulary=None)
```

## Bolsa de Palabras

## Peso = frecuencia del término (Python)

In [3]:

```
print("Tamaño de Vocabulario: {}".format(len(vect.vocabulary_)))
print("Contenido del Vocabulario: {}".format(vect.vocabulary_))
print("Features: {}".format(vect.get_feature_names()))
```

## Bolsa de Palabras

## Peso = frecuencia del término (Python)

In [3]:

```
print("Tamaño de Vocabulario: {}".format(len(vect.vocabulary_)))
print("Contenido del Vocabulario: {}".format(vect.vocabulary_))
print("Features: {}".format(vect.get_feature_names()))
```

Tamaño de Vocabulario: 16

Contenido del Vocabulario: {'pintaron': 9, 'el': 4, 'banco': 1, 'de': 2, 'la': 6, 'plaza': 10, 'te': 15, 'paso': 8, 'programa': 12, 'ejecútalo': 3, 'por': 11, 'sentado': 13, 'en': 5, 'miraba': 7, 'si': 14, 'abría': 0}

Features: ['abría', 'banco', 'de', 'ejecútalo', 'el', 'en', 'la', 'miraba', 'paso', 'pintaron', 'plaza', 'por', 'programa', 'sentado', 'si', 'te']

Bolsa de Palabras

## Transformando los documentos en vectores (Python)

In [4]:

```
bolsa_de_palabras = vect.transform(documentos)
print("Bolsa de palabras: {}\n".format(bolsa_de_palabras))
```

Bolsa de Palabras

## Transformando los documentos en vectores (Python)

In [4]:

```
bolsa_de_palabras = vect.transform(documentos)
print("Bolsa de palabras: {}\n".format(bolsa_de_palabras))
```

Bolsa de palabras: (0, 1) 1

(0, 2) 1

(0, 4) 1

(0, 6) 1

(0, 9) 1

(0, 10) 1

(1, 3) 1

(1, 4) 1

(1, 8) 3

(1, 11) 1

(1, 12) 1

(1, 15) 1

(2, 0) 1

(2, 1) 2

(2, 4) 2

(2, 5) 1

(2, 7) 1

(2, 13) 1

(2, 14) 1

Bolsa de Palabras

## Visualizando la matriz documentos - términos (Python)

In [5] :

```
print("Bolsa de palabras: {}".format(repr(bolsa_de_palabras)))
```

Bolsa de Palabras

## Visualizando la matriz documentos - términos (Python)

In [5]:

```
print("Bolsa de palabras: {}".format(repr(bolsa_de_palabras)))
```

```
Bolsa de palabras: <3x16 sparse matrix of type '<class 'numpy.int64'>'>
with 19 stored elements in Compressed Sparse Row format>
```

Bolsa de Palabras

## Visualizando la matriz documentos - términos (Python)

In [5]:

```
print("Bolsa de palabras: {}".format(repr(bolsa_de_palabras)))
```

```
Bolsa de palabras: <3x16 sparse matrix of type '<class 'numpy.int64'>'>
with 19 stored elements in Compressed Sparse Row format>
```

In [6]:

```
print("Matriz documentos - términos:\n{}".
      format(bolsa_de_palabras.toarray()))
```

Bolsa de Palabras

## Visualizando la matriz documentos - términos (Python)

In [5]:

```
print("Bolsa de palabras: {}".format(repr(bolsa_de_palabras)))
```

```
Bolsa de palabras: <3x16 sparse matrix of type '<class 'numpy.int64'>'>
with 19 stored elements in Compressed Sparse Row format>
```

In [6]:

```
print("Matriz documentos - términos:\n{}".
      format(bolsa_de_palabras.toarray()))
```

```
Matriz documentos - términos:
```

```
[[0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0]
 [0 0 0 1 1 0 0 0 3 0 0 1 1 0 0 1]
 [1 2 0 0 2 1 0 1 0 0 0 0 0 1 1 0]]
```

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

### Documentos

- ① "pintaron el banco de la plaza"
- ② "te paso el programa, ejecútalo paso por paso"
- ③ "sentado en el banco, miraba si el banco abría"

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios (Representación BOW)

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

### Pesos binarios

ID	abría	banco	de	ejecútalo	el	en	la	miraba	paso	pintaron	plaza	por	programa	sentado	si	te
d1	0	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
d2	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1
d3	1	1	0	0	1	1	0	1	0	0	0	0	0	1	1	0

Bolsa de Palabras

## Matriz documentos - términos: pesos binarios (Python)

In [7]:

```
vect_bin = CountVectorizer(binary=True)
vect_bin.fit(documentos)
bow_bin= vect_bin.transform(documentos)
print("Matriz documentos - términos (pesado binario):\n{}".
      format(bow_bin.toarray()))
```

## Matriz documentos - términos: pesos binarios (Python)

In [7]:

```
vect_bin = CountVectorizer(binary=True)
vect_bin.fit(documentos)
bow_bin= vect_bin.transform(documentos)
print("Matriz documentos - términos (pesado binario):\n{}".
format(bow_bin.toarray()))
```

Matriz documentos - términos (pesado binario):

```
[[0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0]
 [0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 1]
 [1 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0]]
```

## Ponderación de términos: ideas principales

- La **importancia (peso)** de un término se **incrementa** proporcionalmente al número de veces que aparece en el documento (supuesto de la **frecuencia del término**)

## Ponderación de términos: ideas principales

- La **importancia (peso)** de un término se **incrementa** proporcionalmente al número de veces que aparece en el documento (supuesto de la **frecuencia del término**)
  - Ayuda a **describir** el contenido del documento.

## Bolsa de Palabras

## Ponderación de términos: ideas principales

- La **importancia (peso)** de un término se **incrementa** proporcionalmente al número de veces que aparece en el documento (supuesto de la **frecuencia del término**)
  - Ayuda a **describir** el contenido del documento.
- La **importancia** general de un término se **decrementa** proporcionalmente a sus ocurrencias en la colección entera (supuesto de la **frecuencia de documento inversa**)

Bolsa de Palabras

## Ponderación de términos: ideas principales

- La **importancia (peso)** de un término se **incrementa** proporcionalmente al número de veces que aparece en el documento (supuesto de la **frecuencia del término**)
  - Ayuda a **describir** el contenido del documento.
- La **importancia** general de un término se **decrementa** proporcionalmente a sus ocurrencias en la colección entera (supuesto de la **frecuencia de documento inversa**)
  - Términos comunes no son buenos para **discriminar** entre clases diferentes.

Bolsa de Palabras

## Ponderación de términos: ideas principales

- La **importancia (peso)** de un término se **incrementa** proporcionalmente al número de veces que aparece en el documento (supuesto de la **frecuencia del término**)
  - Ayuda a **describir** el contenido del documento.
- La **importancia** general de un término se **decrementa** proporcionalmente a sus ocurrencias en la colección entera (supuesto de la **frecuencia de documento inversa**)
  - Términos comunes no son buenos para **discriminar** entre clases diferentes.
- Pesados tipo  $tf - idf$  favorecen **documentos largos** y deberían ser **normalizados**

## Ponderación de términos: enfoques principales

### Pesos binarios

$w_{ji} = 1 \Leftrightarrow$  el documento  $d_j$  contiene el término  $t_i$ , 0 en otro caso.

## Bolsa de Palabras

## Ponderación de términos: enfoques principales

### Pesos binarios

$w_{ji} = 1 \Leftrightarrow$  el documento  $d_j$  contiene el término  $t_i$ , 0 en otro caso.

### Frecuencia del término ( $tf$ )

$w_{ji} = tf(t_i, d_j)$  (número de ocurrencias del término  $t_i$  en el documento  $d_j$ )

Bolsa de Palabras

## Ponderación de términos: enfoques principales

### Pesos binarios

$w_{ji} = 1 \Leftrightarrow$  el documento  $d_j$  contiene el término  $t_i$ , 0 en otro caso.

### Frecuencia del término ( $tf$ )

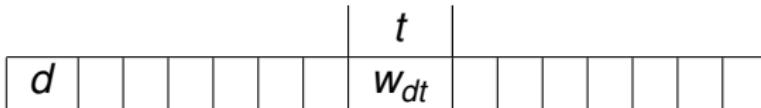
$w_{ji} = tf(t_i, d_j)$  (número de ocurrencias del término  $t_i$  en el documento  $d_j$ )

### Esquema de ponderación $tf \times idf$

$w_{ji} = tf(t_i, d_j) \times idf(t_i)$

- $n$ : número de documentos en la colección
- $idf(t_i) = \log[n/df(t_i)]$ , donde  $df(t_i)$  es el número de documentos que contienen al término  $t_i$

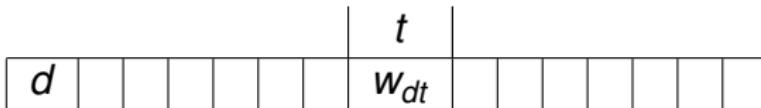
## Notación de ponderación genérica: codificaciones SMART



$$w_{dt} = TF'_{t,d} \cdot IDF'_t \cdot NORM$$

Bolsa de Palabras

## Notación de ponderación genérica: codificaciones SMART



$$w_{dt} = TF'_{t,d} \cdot IDF'_t \cdot NORM$$

### Frecuencia del término

$$n = TF_{t,d}$$

$$b = 1$$

$$m = \frac{TF_{t,d}}{\max_t(TF_{t,d})}$$

$$a =$$

$$0,5 + 0,5 \frac{TF_{t,d}}{\max_t(TF_{t,d})}$$

$$l = 1 + \log(TF_{t,d})$$

### Frecuencia de Documento Inversa

$$n = 1$$

$$t = \log\left(\frac{n}{DF_t}\right)$$

### NORM

$$n = 1$$

$$c = \frac{1}{\sqrt{\sum_t (TF'_{t,d} \cdot IDF'_t)^2}}$$

## Bolsa de Palabras

## Representación BOW - pesado binario

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

Bolsa de Palabras

## Representación BOW - pesado binario

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

### Pesos binarios (SMART bnn)

ID	abría	banco	de	ejecútalo	el	en	la	miraba	paso	pintaron	plaza	por	programa	sentado	si	te
d1	0	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
d2	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1
d3	1	1	0	0	1	1	0	1	0	0	0	0	0	1	1	0

## Bolsa de Palabras

Representación BOW - ponderación *TF*

## Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

Pesos *TF* (Frecuencia del término - SMART nnn)

ID	abría	banco	de	ejecútalo	el	en	la	miraba	paso	pintaron	plaza	por	programa	sentado	si	te
d1	0	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
d2	0	0	0	1	1	0	0	0	3	0	0	1	1	0	0	1
d3	1	2	0	0	2	1	0	1	0	0	0	0	0	1	1	0

Bolsa de Palabras

## Representación BOW - ponderación $tf - idf$ (normalizada)

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

Pesos  $tf - idf$  normalizada (**SMART ntc**)

ID	abría	banco	de	ejecútalo	el	en	la	miraba	paso	pintaron	plaza	por	programa	sentado	si	te
d1	0.	0.34	0.45	0.	0.26	0.	0.45	0.	0.	0.45	0.45	0.	0.	0.	0.	0.
d2	0.	0.	0.	0.27	0.16	0.	0.	0.	0.82	0.	0.	0.27	0.27	0.	0.	0.27
d3	0.33	0.51	0.	0.	0.40	0.33	0.	0.33	0.	0.	0.	0.	0.	0.33	0.33	0.

## Bolsa de Palabras

## Peso = tf-idf normalizada (Python)

In [8]:

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vect_tf_idf = TfidfVectorizer()  
vect_tf_idf.fit(documentos)
```

Bolsa de Palabras

## Peso = tf-idf normalizada (Python)

In [8]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vect_tf_idf = TfidfVectorizer()
vect_tf_idf.fit(documentos)

TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\\\w\\\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

## Bolsa de Palabras

## Peso = tf-idf normalizada (Python)

In [9]:

```
bow_tf_idf_1 = vect_tf_idf.transform(documentos)
print("Matriz documentos - términos (pesado tf-idf por defecto):\n{}".
      format(bow_tf_idf_1.toarray()))
```

Bolsa de Palabras

## Peso = tf-idf normalizada (Python)

In [9]:

```
bow_tf_idf_1 = vect_tf_idf.transform(documentos)
print("Matriz documentos - términos (pesado tf-idf por defecto):\n{}".
      format(bow_tf_idf_1.toarray()))
```

```
Matriz documentos - términos (pesado tf-idf por defecto):
[[0.          0.34261996 0.45050407 0.          0.26607496 0.
  0.45050407 0.          0.          0.45050407 0.45050407 0.
  0.          0.          0.          0.          0.          ],
 [0.          0.          0.          0.2737023 0.16165299 0.
  0.          0.          0.82110689 0.          0.          0.2737023
  0.2737023 0.          0.          0.2737023 ],
 [0.33885833 0.51542099 0.          0.          0.40027038 0.33885833
  0.          0.33885833 0.          0.          0.          ],
 [0.          0.33885833 0.33885833 0.          0.          0.        ]]
```

## Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios - $n$ -gramas (palabras)

### Documentos

- ① "pintaron el banco de la plaza"
- ② "te paso el programa, ejecútalo paso por paso"
- ③ "sentado en el banco, miraba si el banco abría"

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios - $n$ -gramas (palabras)

### Documentos

- ① "pintaron el banco de la plaza"
- ② "te paso el programa, ejecútalo paso por paso"
- ③ "sentado en el banco, miraba si el banco abría"

### bi-gramas de palabras

ID	banco abría	banco de	banco miraba	de la	ejecútalo paso	el banco	el programa	en el	la plaza	...
d1	0	1	0	1	0	1	0	0	1	...
d2	0	0	0	0	1	0	1	0	0	...
d3	1	0	1	0	0	2	0	1	0	...

Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios - $n$ -gramas (caracteres)

### Documentos

- ① "pintaron el banco de la plaza"
- ② "te paso el programa, ejecútalo paso por paso"
- ③ "sentado en el banco, miraba si el banco abría"

## Bolsa de Palabras

## Ejemplo de características dinámicas: Textos arbitrarios - $n$ -gramas (caracteres)

### Documentos

- 1 "pintaron el banco de la plaza"
- 2 "te paso el programa, ejecútalo paso por paso"
- 3 "sentado en el banco, miraba si el banco abría"

### tri-gramas de caracteres

ID	_ab	_ba	_de	_ej	_el	...	aso	aza	ba_	ban	brí	...	tad	tal	tar	te_	úta
d1	0	1	1	0	1	...	0	1	0	1	0	...	0	0	1	0	0
d2	0	0	0	1	1	...	1	0	0	0	0	...	0	1	0	1	1
d3	1	1	0	0	1	...	0	0	1	1	1	...	1	0	0	0	0

## Consideraciones finales sobre BoW

### Ideas subyacentes

- No se captura ningún tipo de información sobre el **orden** en que aparecen los términos/palabras

## Consideraciones finales sobre BoW

### Ideas subyacentes

- No se captura ningún tipo de información sobre el **orden** en que aparecen los términos/palabras
- BoW sólo mira a la **forma superficial** de las palabras, ignorando toda **información semántica** de las mismas

Bolsa de Palabras

## Consideraciones finales sobre BoW

### Ideas subyacentes

- No se captura ningún tipo de información sobre el **orden** en que aparecen los términos/palabras
- BoW sólo mira a la **forma superficial** de las palabras, ignorando toda **información semántica** de las mismas

### Ventajas

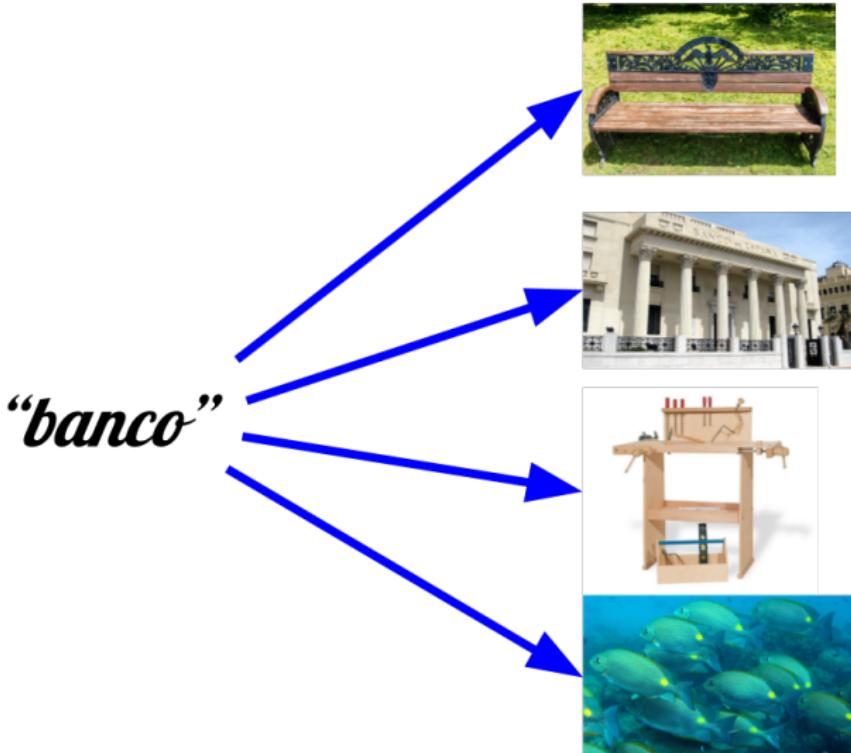
- Simplicidad.
- Eficiencia.

### Desventajas

- BoW tiende a producir **representaciones muy dispersas** (“sparse”)
- Problemas “semánticos” con la **polisemia** y la **sinonimia**

Bolsa de Palabras

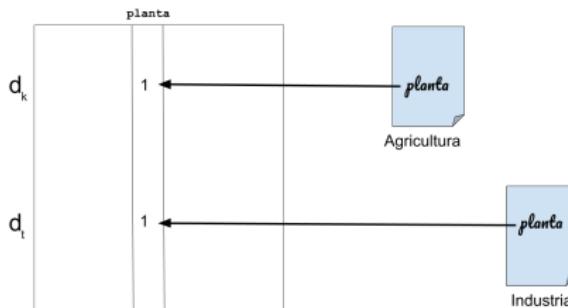
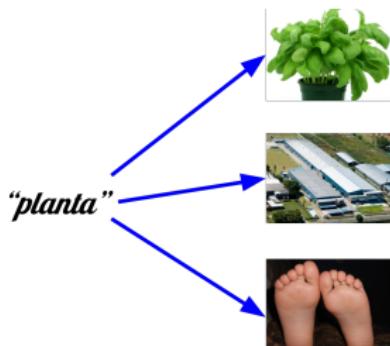
## Polisemia (y homonimia)



Bolsa de Palabras

## BoW y la polisemía

La polisemía introduce ruido en la representación BoW



Bolsa de Palabras

## Sinonimia

***“automotor”***

***“auto”***

***“coche”***

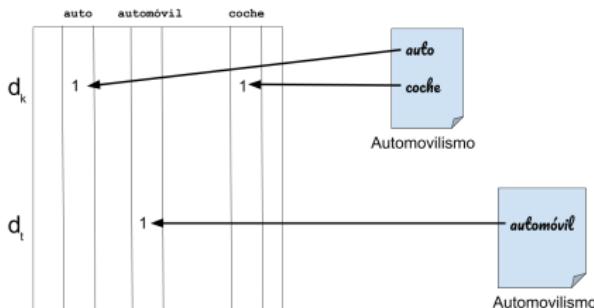
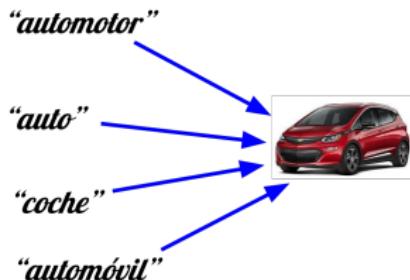
***“automóvil”***



Bolsa de Palabras

## BoW y la sinonimia

La **sinonimia** divide la evidencia en la **representación BoW**



## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.

## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
  - **Circularidad** de las definiciones de diccionario.

## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
  - Circularidad de las definiciones de diccionario.
  - Dificultad para capturar otros tipos de **relaciones** entre palabras (más alla de las relaciones semanticas clásicas de WN) (**asociación** de palabras, **campos semánticos**, **significados afectivos/connotaciones**).

## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
  - Circularidad de las definiciones de diccionario.
  - Dificultad para capturar otros tipos de **relaciones** entre palabras (más alla de las relaciones semanticas clásicas de WN) (**asociación** de palabras, **campos semánticos**, **significados afectivos/connotaciones**).
- Identificadas por filósofos como **Ludwig Wittgenstein**:  
*... the **meaning** of a word is its **use** in the language*

## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
  - Circularidad de las definiciones de diccionario.
  - Dificultad para capturar otros tipos de **relaciones** entre palabras (más alla de las relaciones semanticas clásicas de WN) (**asociación** de palabras, **campos semánticos**, **significados afectivos/connotaciones**).
- Identificadas por filósofos como **Ludwig Wittgenstein**:  
*... the **meaning** of a word is its **use** in the language*
- ... y lingüistas como **John R. Firth**:  
*You shall **know** a word by the **company** it keeps!*

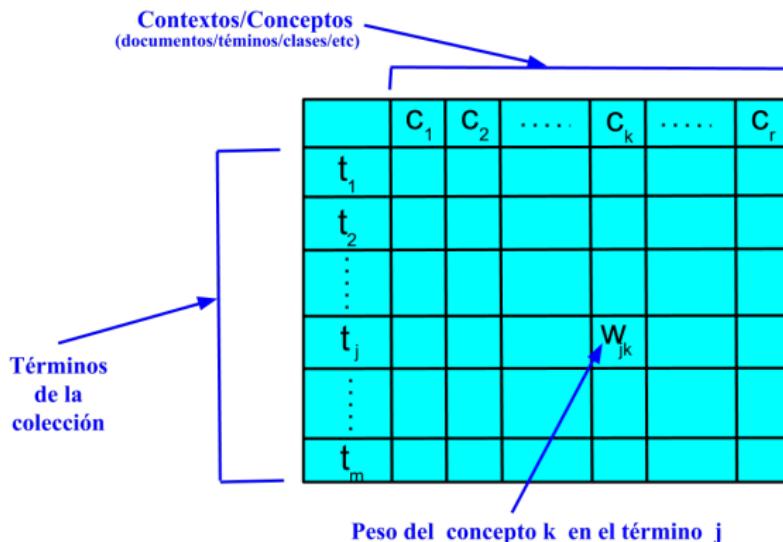
## Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
  - Circularidad de las definiciones de diccionario.
  - Dificultad para capturar otros tipos de **relaciones** entre palabras (más alla de las relaciones semanticas clásicas de WN) (**asociación** de palabras, **campos semánticos**, **significados afectivos/connotaciones**).
- Identificadas por filósofos como **Ludwig Wittgenstein**:  
*... the **meaning** of a word is its **use** in the language*
- ... y lingüistas como **John R. Firth**:  
*You shall **know** a word by the **company** it keeps!*

## Enfoques Distribucionales

## Representación distribucional de términos

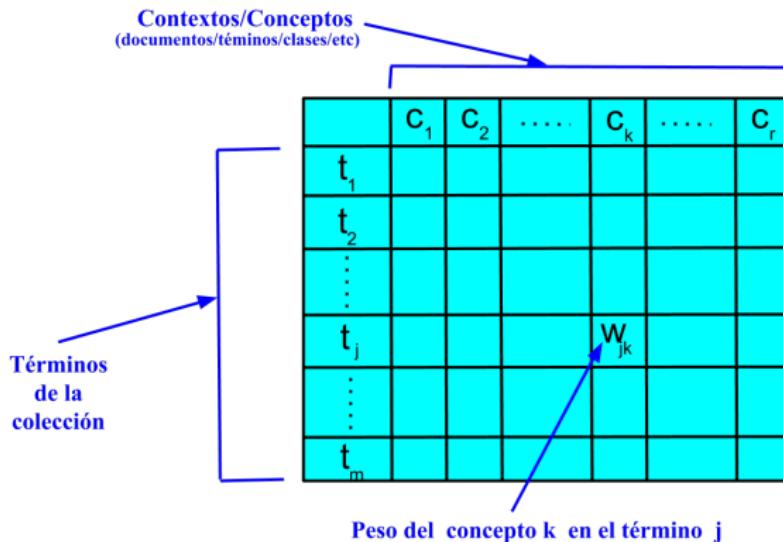
- Propuesto para abordar las **deficiencias** del modelo BoW
- El foco se pone en las **palabras** y los **contextos** en que éstas ocurren



## Enfoques Distribucionales

## Representación distribucional de términos

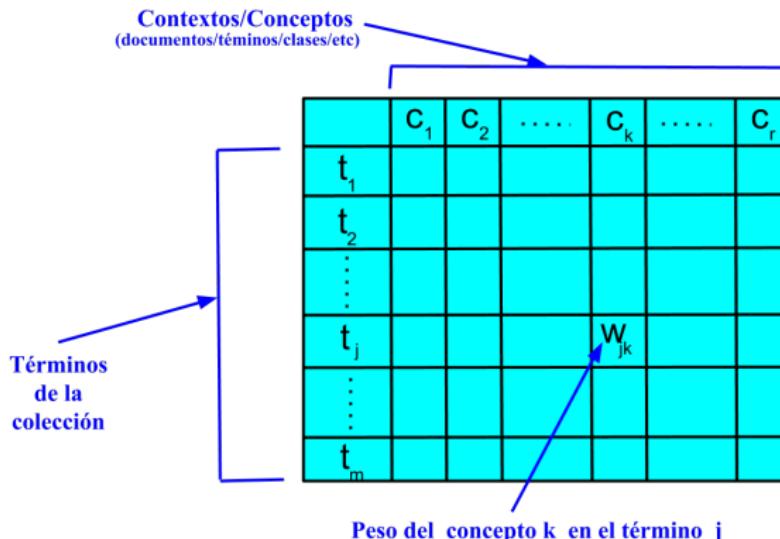
- Se cambia el foco en medir **similitud de palabras**
- **Hipótesis distribucional:** palabras que ocurren en contextos similares tienden a tener **significados similares.**



## Enfoques Distribucionales

## Representación distribucional de términos

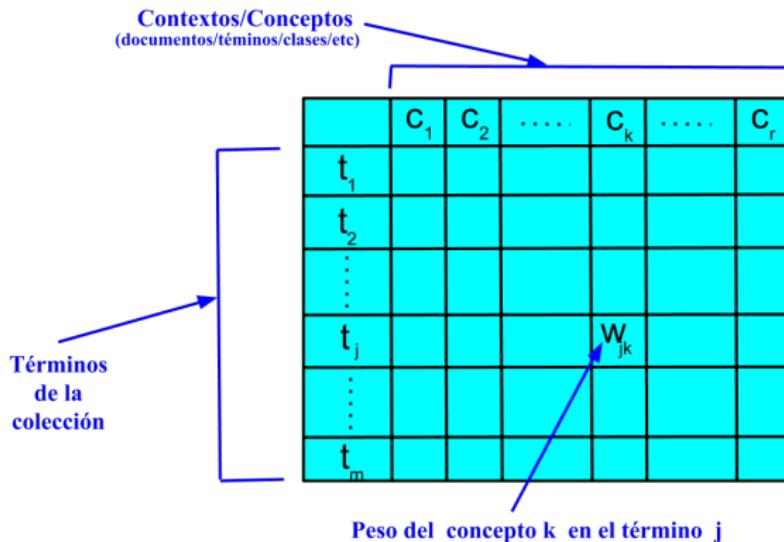
- Cada palabra es representada por un **vector**
- Cada elemento del vector se deriva de la **ocurrencia de la palabra en distintos contextos**: otras **palabras, frases, sentencias, párrafos, capítulos, documentos**, etc.



## Enfoques Distribucionales

# Representación distribucional de términos

- Cada **contexto** puede ser visualizado como un **concepto**
- Cada fila de la matriz palabra-concepto puede ser visualizada entonces como una **bolsa de conceptos (BOC)**



## Algunas representaciones distribucionales conocidas

Existen distintas variantes de las representaciones distribucionales.

## Algunas representaciones distribucionales conocidas

Existen distintas variantes de las representaciones distribucionales.

- Document occurrence representation (**DOR**)
- Term co-occurrence representation (**TCOR**)
- Concise semantic analysis (**CSA**)

Veremos ahora, el caso en que estas **representaciones de los términos** son **aprendidas**

## Características aprendidas

- ➊ Idea: extender el aprendizaje automático, usualmente usado para generar el modelo de clasificación, a la representación de los documentos.

## Características aprendidas

- ① Idea: extender el **aprendizaje automático**, usualmente usado para generar el **modelo de clasificación**, a la **representación de los documentos**.
- ② Componente principal en el **aprendizaje de representaciones** para el PLN: **word embedding (WE)** (incrustación de palabra)

## Características aprendidas

- ① Idea: extender el **aprendizaje automático**, usualmente usado para generar el **modelo de clasificación**, a la **representación de los documentos**.
- ② Componente principal en el **aprendizaje de representaciones** para el PLN: **word embedding (WE)** (incrustación de palabra)
- ③ Los **WEs** son **representaciones distribuidas** de palabras basadas en **vectores densos**, de longitud **fija** construidas mediante estadísticas de co-ocurrencia de palabras según la **hipótesis distribucional**

## Características aprendidas (II)

- ① Los WEs pueden ser derivados mediante enfoques **basados en conteo**:
  - LSA
  - HAL
  - COALS
  - ... y Glove (entre otros)
- ② Sin embargo, se han popularizado últimamente los enfoques **basados en predicción** utilizando métodos de **aprendizaje neuronales**: **word2vec**, **fasttext**, **Elmo** y **Bert**
- ③ Los WEs, además de capturar relaciones **sintácticas** y **semánticas** muy interesantes de las palabras, soportan el funcionamiento de modelos neuronales más generales para documentos como las arquitecturas **LSTM** y **CNN**.

Enfoques Distribucionales

## Word2vec

word2vec [Mikolov et. al]



Disponible en:

<https://code.google.com/archive/p/word2vec/>

Enfoques Distribucionales

## Word2vec

word2vec [Mikolov et. al]



Disponible en:

<https://code.google.com/archive/p/word2vec/>

Enfoques Distribucionales

## Word2vec

- El método **más popular** de **embedding** en la actualidad.

## Enfoques Distribucionales

## Word2vec

- El método **más popular** de **embedding** en la actualidad.
- A diferencia de los enfoques **distribucionales**, la idea es **predecir** más que **contar**

## Enfoques Distribucionales

## Word2vec

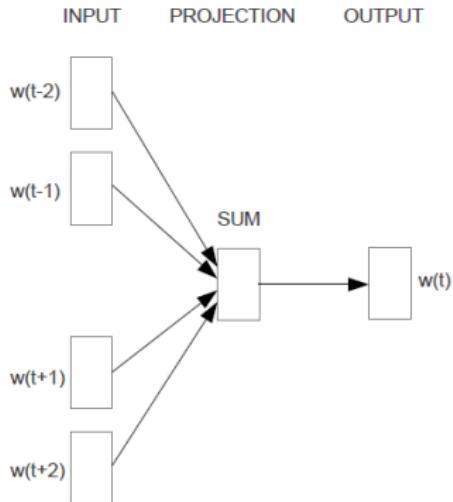
- El método **más popular** de **embedding** en la actualidad.
- A diferencia de los enfoques **distribucionales**, la idea es **predecir** más que **contar**
- Se **aprende** un **predictor** (**clasificador**), y como **efecto colateral** se obtienen los “**embeddings**” (**vectores**) que representan las palabras.

## Word2vec

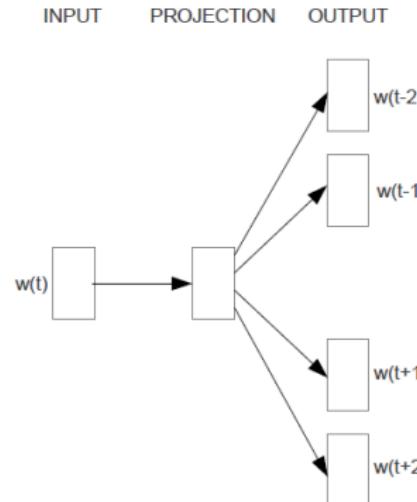
- El método **más popular** de **embedding** en la actualidad.
- A diferencia de los enfoques **distribucionales**, la idea es **predecir** más que **contar**
- Se **aprende** un **predictor (clasificador)**, y como **efecto colateral** se obtienen los “**embeddings**” (**vectores**) que representan las palabras.
- Enfoque bastante **eficiente** para aprender

## Enfoques Distribucionales

## Enfoques en Word2vec



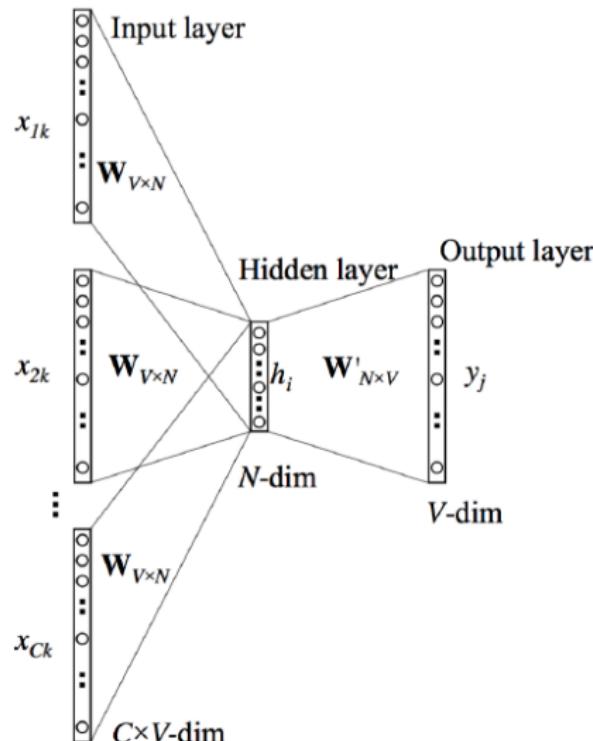
CBOW



Skip-gram

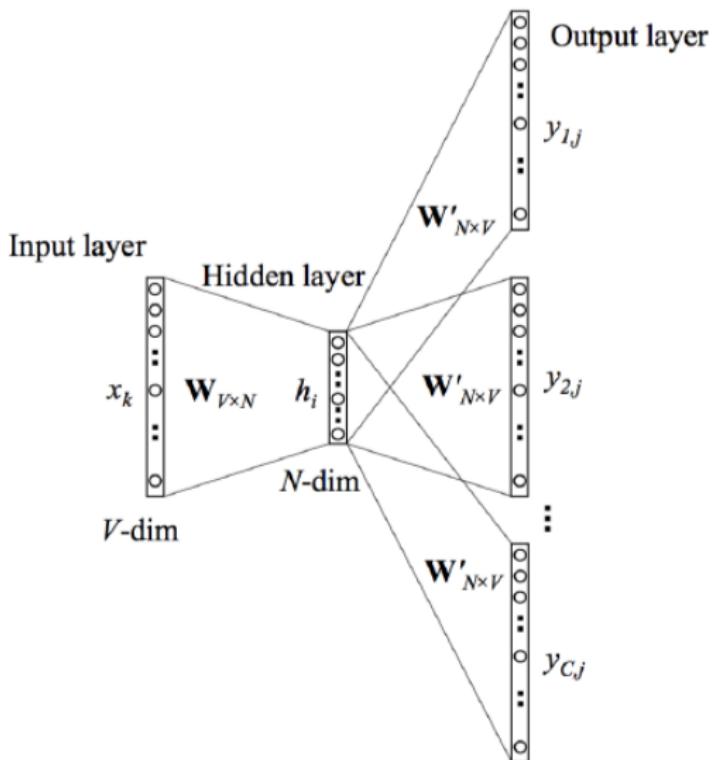
## Enfoques Distribucionales

## Detalles de CBOW



## Enfoques Distribucionales

## Detalles de Skip-gram



## Reducción de la dimensionalidad

- Problema:

## Reducción de la dimensionalidad

- **Problema:** el conjunto de términos  $\mathcal{T}$  que ocurre al menos una vez en una colección de documentos puede ser **enorme**.

## Reducción de la dimensionalidad

- **Problema:** el conjunto de términos  $\mathcal{T}$  que ocurre al menos una vez en una colección de documentos puede ser **enorme**.
- Esto implica que en una representación vectorial, debamos tratar con dimensiones de vectores en el orden de los **miles** de atributos (decenas o cientos).

## Reducción de la dimensionalidad

- **Problema:** el conjunto de términos  $\mathcal{T}$  que ocurre al menos una vez en una colección de documentos puede ser **enorme**.
- Esto implica que en una representación vectorial, debamos tratar con dimensiones de vectores en el orden de los **miles** de atributos (decenas o cientos).
- **Solución:**

## Reducción de la dimensionalidad

- **Problema:** el conjunto de términos  $\mathcal{T}$  que ocurre al menos una vez en una colección de documentos puede ser **enorme**.
- Esto implica que en una representación vectorial, debamos tratar con dimensiones de vectores en el orden de los **miles** de atributos (decenas o cientos).
- **Solución:** convertir  $\mathcal{T}$  a otro espacio de términos  $\mathcal{T}'$ , tal que  $|\mathcal{T}'| \ll |\mathcal{T}|$

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.
- El **tiempo** que insume entrenar un clasificador también es menor que con el conjunto de términos original.

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.
- El **tiempo** que insume entrenar un clasificador también es menor que con el conjunto de términos original.
- En algunos casos, la reducción de términos permite eliminar palabras **redundantes** o que sólo introducen **ruido** en el aprendizaje del clasificador.

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.
- El **tiempo** que insume entrenar un clasificador también es menor que con el conjunto de términos original.
- En algunos casos, la reducción de términos permite eliminar palabras **redundantes** o que sólo introducen **ruido** en el aprendizaje del clasificador.
- La eliminación de estas palabras puede resultar en un clasificador **más efectivo** (más exacto a la hora de clasificar) que el que se obtenía con el conjunto de términos orginal.

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.
- El **tiempo** que insume entrenar un clasificador también es menor que con el conjunto de términos original.
- En algunos casos, la reducción de términos permite eliminar palabras **redundantes** o que sólo introducen **ruido** en el aprendizaje del clasificador.
- La eliminación de estas palabras puede resultar en un clasificador **más efectivo** (más exacto a la hora de clasificar) que el que se obtenía con el conjunto de términos orginal.

## Enfoques para reducir la dimensionalidad

### **Selección de términos**

Sólo se mantienen los términos **más relevantes** ( $T' \subseteq T$ ) de acuerdo con alguna **función de valoración** como la **ganancia de información** o **Chi-cuadrado** ( $\chi^2$ ).

### **Transformación del espacio de términos**

$T'$  no es un subconjunto de  $T$ , e incluso pueden ser atributos de un **tipo completa/ distinto**. Ejemplo: **LSA** por **Latent Semantic Analysis**

Indexado (análisis) de semántica latente

## LSA - Idea:

Descomponer una matriz  $X$  de  $n$  documentos  $\times$   $m$  términos:

Indexado (análisis) de semántica latente

## LSA - Idea:

Descomponer una matriz  $X$  de  $n$  documentos  $\times$   $m$  términos:

**X**

	$t_1$	$t_2$	.....	$t_k$	.....	$t_m$
$d_1$						
$d_2$						
⋮						
$d_j$						
⋮						
$d_n$						

$$|D| \times |T|$$

Indexado (análisis) de semántica latente

**LSA - Idea:**

Descomponer una matriz  $X$  de  $n$  documentos  $\times m$  términos:

	$t_1$	$t_2$	.....	$t_k$	.....	$t_m$
$d_1$						
$d_2$						
⋮						
$d_j$						
⋮						
$d_n$						

$$|D| \times |T|$$

En 3 matrices

- $U$  de  $n$  documentos  $\times k$  conceptos
- $S$  de  $k$  conceptos  $\times k$  conceptos
- $V$  de  $m$  términos  $\times k$  conceptos

Indexado (análisis) de semántica latente

**LSA - Idea:**

Descomponer una matriz  $X$  de  $n$  documentos  $\times m$  términos:

		$t_1$	$t_2$	.....	$t_k$	.....	$t_m$
$d_1$							
$d_2$							
⋮							
$d_j$							
⋮							
$d_n$							

$|D| \times |T|$

En 3 matrices

$U$	$S$	$V$
$d_1$	$q_1$	$t_1$
$d_2$	$q_2$	$t_2$
⋮	⋮	⋮
$d_i$	$q_i$	$t_i$
⋮	⋮	⋮
$d_n$	$q_n$	$t_n$

$|D| \times k$        $k \times k$        $|T| \times k$

## Indexado (análisis) de semántica latente

**tal que ...:**

	$t_1$	$t_2$	.....	$t_i$	.....	$t_m$
$d_1$						
$d_2$						
.....						
$d_j$						
.....						
$d_n$						

The diagram shows the decomposition of a matrix  $X$  into three components:  $U_k$ ,  $S_k$ , and  $V_k^T$ .

- $U_k$  is a vertical matrix with  $D$  rows and  $k$  columns. The columns are labeled  $c_1, c_2, \dots, c_k$ . The rows are labeled  $d_1, d_2, \vdots, d_j, \vdots, d_n$ .
- $S_k$  is a square matrix of size  $k \times k$ . The columns are labeled  $c_1, c_2, \dots, c_k$ . The rows are labeled  $c_1, c_2, \vdots, c_k$ .
- $V_k^T$  is a horizontal matrix with  $k$  columns and  $|T|$  rows. The columns are labeled  $t_1, t_2, \dots, t_k, \dots, t_m$ . The rows are labeled  $c_1, c_2, \vdots, c_k$ .

Indexado (análisis) de semántica latente

## Indexado (análisis) de semántica latente

Idea:

## Indexado (análisis) de semántica latente

## Indexado (análisis) de semántica latente

Idea:

- Descomponer (factorizar) una matriz  $X$  de  $n$  documentos  $\times m$  términos en 3 matrices:
  - $U$  de  $n$  documentos  $\times k$  conceptos
  - $S$  de  $k$  conceptos  $\times k$  conceptos
  - $V$  de  $m$  términos  $\times k$  conceptos

## Indexado (análisis) de semántica latente

## Indexado (análisis) de semántica latente

Idea:

- Descomponer (factorizar) una matriz  $X$  de  $n$  documentos  $\times m$  términos en 3 matrices:
  - $U$  de  $n$  documentos  $\times k$  conceptos
  - $S$  de  $k$  conceptos  $\times k$  conceptos
  - $V$  de  $m$  términos  $\times k$  conceptos
- tal que:

$$X \approx USV^T$$

## Indexado (análisis) de semántica latente

# Indexado (análisis) de semántica latente

Idea:

- $U \times S$  es el embedding de los documentos <—
- $V \times S$  es el embedding de los términos

$U_k$

	$c_1$	$c_2$	.....	$c_k$
$d_1$				
$d_2$				
⋮				
$d_j$				
⋮				
$d_n$				

$S_k$

	$c_1$	$c_2$	...	$c_k$
$c_1$				
$c_2$				
⋮				
$c_k$				

$k \times k$

$X$

$|D| \times k$

## Indexado (análisis) de semántica latente

# Indexado (análisis) de semántica latente

Idea:

- $U \times S$  es el embedding de los documentos
- $V \times S$  es el embedding de los términos <—

$V_k$

	$c_1$	$c_2$	.....	$c_k$
$t_1$				
$t_2$				
:				
$t_j$				
:				
$t_m$				

$S_k$

	$c_1$	$c_2$	.....	$c_k$
$c_1$				
$c_2$				
:				
$c_k$				

$k \times k$

$X$

$|T| \times k$

## Indexado (análisis) de semántica latente

## Indexado (análisis) de semántica latente

Idea:

- $U \times S$  es el embedding de los documentos
- $P \times S$  es el embedding de los términos

Un documento  $\vec{d}$  de *test m-dimensional* es fácilmente convertido a un vector *k-dimensional* haciendo  $\vec{d} \times V$

## Indexado (análisis) de semántica latente

# Indexado (análisis) de semántica latente

## Consideraciones generales

- Es la aplicación del método **singular value decomposition (SVD)** a datos textuales

