

# Analítica de Textos

Manuel Sánchez-Montañés

# Analítica de Textos

---

There is a credit card showing up on my credit report which is not mine. I have never taken out a Capital One credit card. Please help

I have been paying extra money each month on my mortgage. This month I received a check for {\$420.00} as a reimbursement for escrow. My escrow amount was also increased which makes no sense at all. I believe that charging this much escrow for no reason is a ridiculous use of my money. Please help.

I have had issues in the past with overdrafts to my business checking accounts. My bank 's online banking does not allow transfers after XXXX to go through until the next business day ( two days later ), even if I have funds directly in the bank.

...

¿Conocimiento cuantitativo?



# ¿Qué es el procesamiento de lenguaje natural (NLP)?

---

**NLP:** conjunto grande de técnicas para el análisis automático, procesamiento y generación de lenguaje humano

## **Algunas aplicaciones importantes:**

- Traducción automática
- Comprensión de lenguaje natural (por parte de la máquina)
- Generación de lenguaje natural (p.e. resúmenes automáticos, chatbots)
- Respuesta automática a preguntas
- Análisis de sentimiento
- Categorización de documentos / detección automática de temas
- Búsqueda inteligente
- Filtrado de spam
- Reconocimiento de habla
- Generadores automáticos de habla a partir de texto

# ¿Qué es el procesamiento de lenguaje natural (NLP)?

---

## **Aplicaciones en medios sociales (I)**

- Aplicaciones de respuesta a desastres
- Comunicación gubernamental
- Monitorización de medios
- Detección en comunidades online de individuos en riesgo: depresión, prevención de suicidio, monitorización de pacientes (con su consentimiento)
- Economía conductual: medida automática del “estado de ánimo general” (muy útil en aplicaciones financieras ya que está correlacionado con los indicadores económicos)
- Seguridad y defensa (detección de posibles amenazas, individuos que expresan altos niveles de odio pueden estar asociados a amenazas terroristas)

# ¿Qué es el procesamiento de lenguaje natural (NLP)?

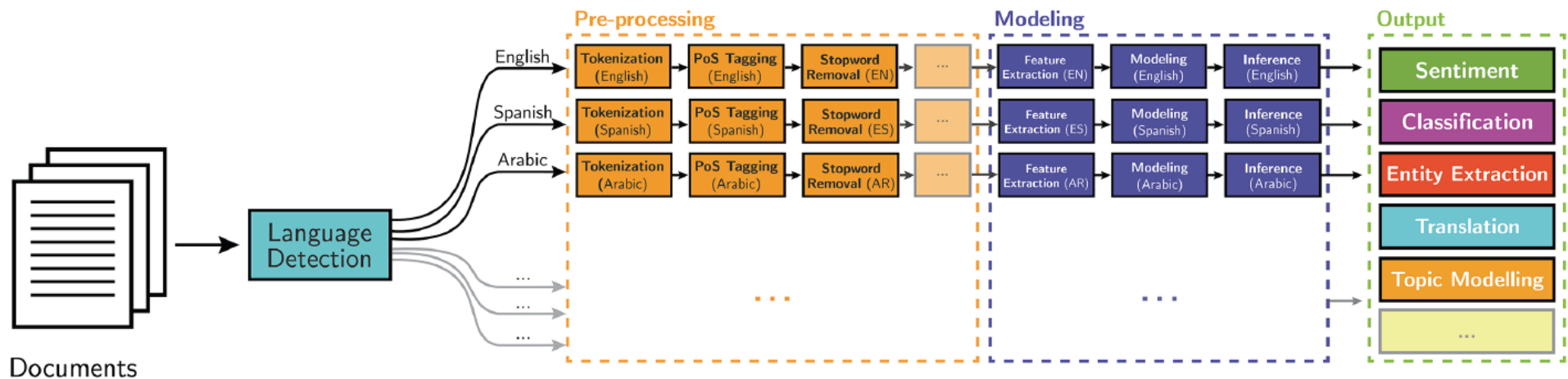
---

## **Aplicaciones en medios sociales (II)**

- Marketing: marketing predictivo, análisis de sentimiento, gestión de campañas, entender tendencias sociales clave.
- Recomendación de productos personalizada basada en los datos de esa persona en medios sociales públicos
- Encuestas en medios sociales

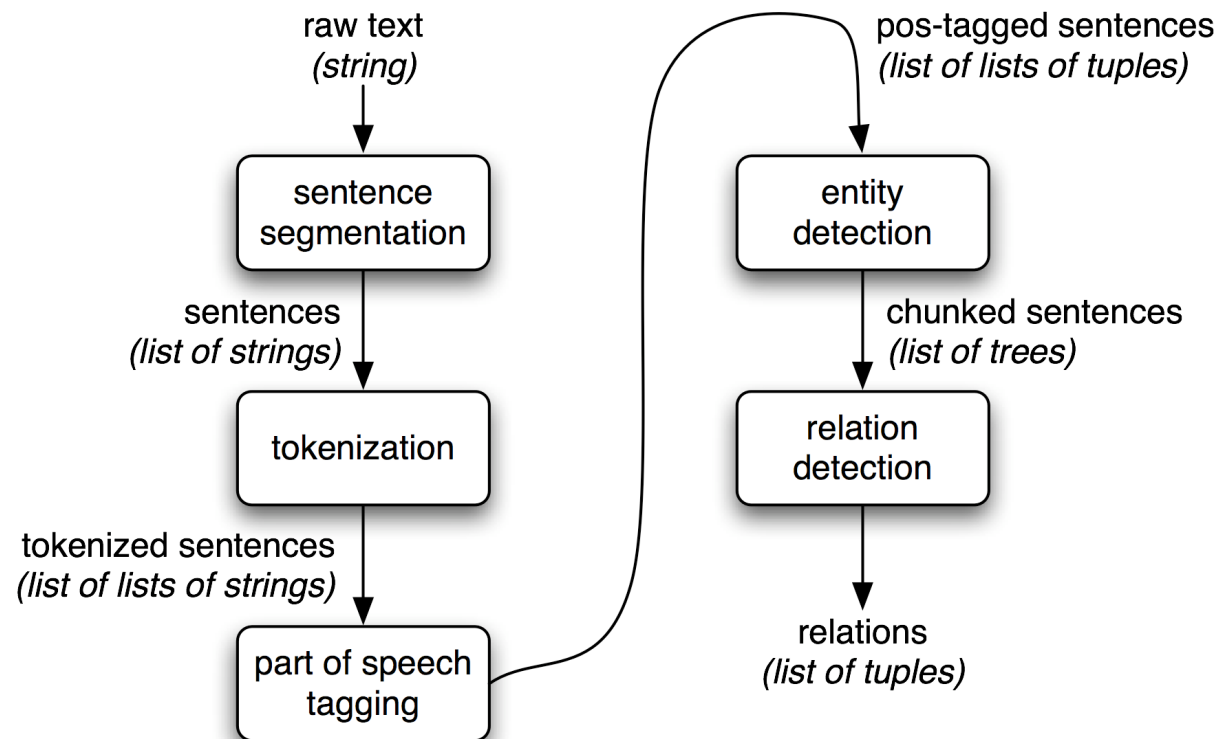
# Esquema del NLP clásico

## NLP clásico



# Estrategias para extraer información en NL

Una arquitectura de extracción de información típica es:



[Algunas figuras de [www.nltk.org](http://www.nltk.org)]

# Estrategias para extraer información en NL

---

## Segmentación de frases:

“We saw the yellow dog. It looked tired and hungry”

⇒ [“We saw the yellow dog”, “It looked tired and hungry”]

## Tokenización:

[“We saw the yellow dog”, “It looked tired and hungry”]

⇒ [ [We, saw, the, yellow, dog], [It, looked, tired, and, hungry ] ]



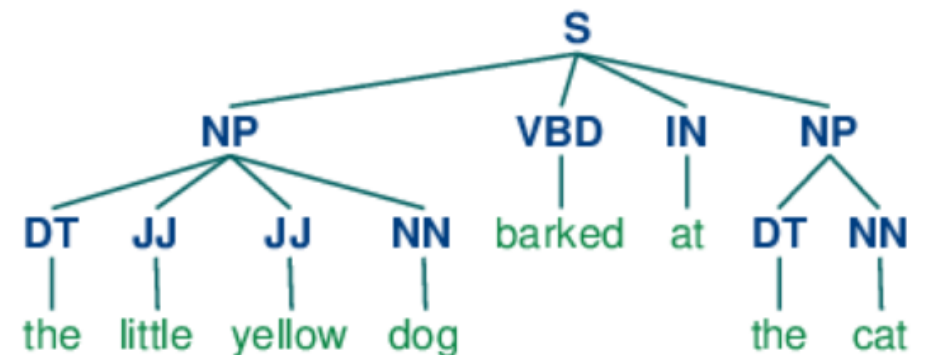
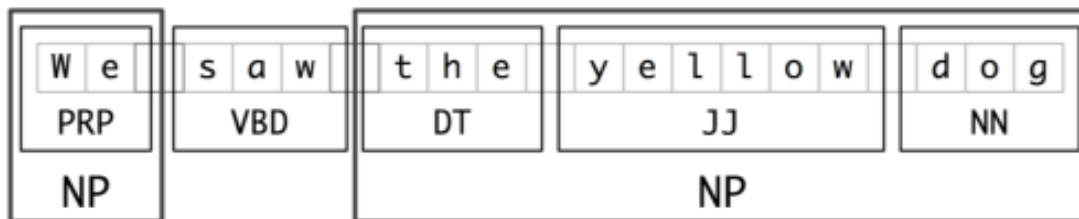
# Estrategias para extraer información en NL

## Pos-tagging:

“We saw the yellow dog”

⇒ [(“We”, “PRP”), (“saw”, “VBD”), (“DT”, “the”), (“yellow”, “JJ”), (“NN”, “dog”)]

## Chunking:



# Etapas de preprocesamiento fundamentales en un proyecto de analítica de textos

---

1. **Preprocesado de textos**
2. **Word / Paragraph / Document vector encoding**
3. **Procesamiento de Lenguaje Natural de Alto Nivel**  
Clustering, construcción de modelos predictivos, etc.

# Parte 1. Preprocesado de Textos

---

- **Lematizador**

“cables”  $\Rightarrow$  “cable”

- **Eliminación de stop words**

Stop word: cualquier palabra que no esperemos que contenga contenido semántico importante.

Podemos cargar una lista de stopwords predefinida en NLTK, cambiarla, o crear nuestra propia lista de cero.

[“hello”, “john”, “bought”, “two”, “cable”, “and”, “lcd”, “monitor”]

$\Rightarrow$  [“hello”, “john”, “bought”, “cable”, “lcd”, “monitor”]

## Parte 2. Vector encoding

---

### Estrategia 1: BOW (Bag of Words)

[ “bought”, “car”, “frequent”, “failure”]	#documento 0
[ “cost”, “bought”, “car”, “low”, “energy”, “low”, “cost”]	# documento 1
[ “failure”, “bought”, “motorcycle”, “low”, “quality”]	# documento 2
[ “frequent”, “low”, “quality”, “bought”, “car”]	# documento 3

Nuestra codificación de los documentos es la matriz “term frequency” (tf) donde:

$tf[i,j]$  = número de apariciones del término  $j$  en el documento  $i$

Cada documento está representado por un vector de tantas componentes como palabras en el vocabulario (puede ser enorme!!)

## Parte 2. Vector encoding

### Estrategia 1. BOW (Bag of Words)

[“bought”, “car”, “frequent”, “failure”]	#documento 0
[“cost”, “bought”, “car”, “low”, “energy”, “low”, “cost”]	# documento 1
[“failure”, “bought”, “motorcycle”, “low”, “quality”]	# documento 2
[“frequent”, “low”, “quality”, “bought”, “car”]	# documento 3

**vocabulario** = [“bought”, “car”, “cost”, “energy”, “failure”, “frequent”, “low”, “motorcycle”, “quality”]


$$tf = \begin{matrix} & \begin{matrix} \text{bought} & \text{car} & \text{cost} & \text{energy} & \text{failure} & \text{frequent} & \text{low} & \text{motorcyc} & \text{quality} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} & \left. \vphantom{\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}} \right\} \begin{matrix} \text{número de} \\ \text{documentos} \end{matrix} \\ \underbrace{\hspace{15em}} & \text{Número de palabras} \end{matrix}$$


## Parte 2. Vector encoding

---

### Estrategia 2. TF-IDF (1)

**Idea:** Un término que aparece en todos los documentos no es muy informativo. Un término que aparece en un subconjunto pequeño (cuidado: demasiado pequeño es también malo) es mucho más informativo.

 **df[t]** (“document frequency”): número de documentos donde el término (palabra) *t* aparece al menos una vez

 **idf[t]** (“inverse of document frequency”): función de *df[t]* que decrece monotónicamente si *df* aumenta. Por ejemplo:

$$idf[t] = 1 + \log \left[ \frac{D + 1}{df[t] + 1} \right]$$

**D:** número total de documentos en el corpus. El “+1” en el numerador y en el denominador evita problemas numéricos:  $\log(0/\text{número})$  o  $\log(\text{número}/0)$ .

Otras variantes muy similares a esta ecuación existen en la literatura y también se conocen como “idf”.

## Parte 2. Vector encoding

---

### Estrategia 2. TF-IDF (2)

Del ejemplo anterior:

$$tf = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

🌐 **vocabulario** = ["bought", "car", "cost", "energy", "failure", "frequent", "low", "motorcycle", "quality"]

🌐 D=4. df = [4, 3, 1, 1, 2, 2, 3, 1, 2]


🌐  $idf[t] = 1 + \log \left[ \frac{D+1}{df[t]+1} \right] = [1, 1.22, 1.92, 1.92, 1.51, 1.51, 1.22, 1.92, 1.51]$

🌐 La matriz TF-IDF se computa multiplicando cada columna t de TF por  $idf[t]$

## Parte 2. Vector encoding


### •Estrategia 2. TF-IDF (3)

$$tf = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

 **vocabulario** = ["bought", "car", "cost", "energy", "failure", "frequent", "low", "motorcycle", "quality"]

$$\text{idf}[t] = 1 + \log \left[ \frac{D+1}{df[t]+1} \right] = [1, 1.22, 1.92, 1.92, 1.51, 1.51, 1.22, 1.92, 1.51]$$

$$tfidf = \begin{bmatrix} 1 & 1.22 & 0 & 0 & 1.51 & 1.51 & 0 & 0 & 0 \\ 1 & 1.22 & 3.84 & 1.92 & 0 & 0 & 2.44 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1.51 & 0 & 1.22 & 1.92 & 1.51 \\ 1 & 1.22 & 0 & 0 & 0 & 1.51 & 1.22 & 0 & 1.51 \end{bmatrix}$$

 Cada fila de la matriz tfidf puede ser normalizada individualmente de acuerdo por ejemplo a la norma L2  $\Rightarrow$  los documentos muy grandes se pueden comparar "en igualdad de condiciones" a los documentos muy pequeños



## Parte 2. Vector encoding

---

### Estrategia 3: Latent Semantic Analysis (LSA), 1

- 🌐 Singular Value Decomposition (SVD): cualquier matriz  $M$  de dimensiones  $D \times W$  ( $D$ : número de documentos;  $W$ : número de palabras) puede descomponerse **exactamente** como

$$M = U \cdot \Sigma \cdot V^T$$

donde:

- $\Sigma$  es una matriz diagonal con componentes no negativos y dimensiones  $L \times L$
- $L$  (“dimensiones del espacio latente”) =  $\min(D, W)$
- $U$  es una matriz de  $D \times L$  dimensiones donde cada columna está normalizada y es ortogonal a las otras
- $V$  es una matriz de  $W \times L$  dimensiones donde cada columna está normalizada y es ortogonal a las otras

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 2

- Ejemplo:

$$M = \begin{bmatrix} [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \\ [0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 1] \\ [1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0] \\ [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \end{bmatrix} = U \cdot \Sigma \cdot V^T$$

con

$$U = \begin{bmatrix} [-0.53609237 & 0.03922979 & 0.4594192 & -0.70710678] \\ [-0.63342998 & -0.29925753 & -0.71359049 & 0.] \\ [-0.15484316 & 0.95255812 & -0.26202407 & 0.] \\ [-0.53609237 & 0.03922979 & 0.4594192 & 0.70710678] \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} [3.87493377 & 0. & 0. & 0.] \\ [0. & 1.94118656 & 0. & 0.] \\ [0. & 0. & 1.79351137 & 0.] \\ [0. & 0. & 0. & 0.] \end{bmatrix} \quad V = \begin{bmatrix} [-0.03996021 & 0.49070921 & -0.14609557 & -0.16990314] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.31889378] \\ [-0.27669756 & 0.04041836 & 0.51231256 & -0.30320033] \\ [-0.44016616 & -0.11374381 & 0.11443914 & 0.8370474] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.32693719 & -0.30832433 & -0.79574683 & -0.15160017] \\ [-0.48012637 & 0.3769654 & -0.03165643 & -0.05200069] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.1629526] \end{bmatrix}$$

## Parte 2. Vector encoding

---

### Estrategia 3: Latent Semantic Analysis (LSA), 3

- Las componentes de  $\Sigma$  se interpretan como las “importancias” de cada dimensión latente y se ordenan en orden decreciente:

$$\Sigma = \begin{bmatrix} [3.87493377 & 0. & 0. & 0. & ] \\ [0. & 1.94118656 & 0. & 0. & ] \\ [0. & 0. & 1.79351137 & 0. & ] \\ [0. & 0. & 0. & 0. & ] \end{bmatrix}$$

- En este caso la cuarta dimensión latente tiene “importancia” nula (0).
- Eliminarla es equivalente a usar las siguientes matrices:

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 4

- Si “eliminamos” la cuarta dimensión latente:

$$U = \begin{bmatrix} [-0.53609237 & 0.03922979 & 0.4594192 & -0.70710678] \\ [-0.63342998 & -0.29925753 & -0.71359049 & 0.] \\ [-0.15484316 & 0.95255812 & -0.26202407 & 0.] \\ [-0.53609237 & 0.03922979 & 0.4594192 & 0.70710678] \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} [3.87493377 & 0. & 0. & 0.] \\ [0. & 1.94118656 & 0. & 0.] \\ [0. & 0. & 1.79351137 & 0.] \\ [0. & 0. & 0. & 0.] \end{bmatrix} \quad V = \begin{bmatrix} [-0.03996021 & 0.49070921 & -0.14609557 & -0.16990314] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.11889178] \\ [-0.27669756 & 0.04041836 & 0.51231256 & -0.30320033] \\ [-0.44016616 & -0.11374381 & 0.11443914 & 0.8370474] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.32693719 & -0.30832433 & -0.79574683 & -0.15100017] \\ [-0.48012637 & 0.3769654 & -0.03165643 & -0.0201069] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.11889178] \end{bmatrix}$$

(esto es, reducimos el número de columna

Si ahora calculamos

obtenemos de nuevo la matriz M original:

$$U \cdot \Sigma \cdot V^T$$

$$\begin{bmatrix} [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \\ [0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 1] \\ [1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0] \\ [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \end{bmatrix}$$

## Parte 2. Vector encoding

---

### Estrategia 3: Latent Semantic Analysis (LSA), 5

- Uso práctico de LSA:

Computar la descomposición SVD de la matriz tfidf

Eliminar las dimensiones latentes con menos información

Eliminar las correspondientes columnas en  $U$ ,  $V$ ,  $\Sigma$

Ahora la fila  $d$  en  $U \cdot \Sigma$  se interpreta como la vectorización del documento  $d$

La fila  $w$  en  $V \cdot \Sigma$  se interpreta como la vectorización de la palabra  $w$

- Por tanto **hemos asignado un vector de dimensión  $L'$  a cada documento y palabra**
- A partir de ahí se construirán modelos de alto nivel con esa vectorización

## Parte 2. Vector encoding

---

### BOW / TF-IDF / LSA:

**Es conveniente normalizar cada documento (por lo que la "longitud" de todos los vectores del documento es 1)**

Esto permite comparar documentos pequeños con documentos grandes

- Hacer esto es equivalente a no normalizar pero usando la "distancia del coseno"

$$\text{Cosine distance}(\mathbf{v1}, \mathbf{v2}) = 1 - \frac{\mathbf{v1} \cdot \mathbf{v2}}{\|\mathbf{v1}\| \|\mathbf{v2}\|}$$

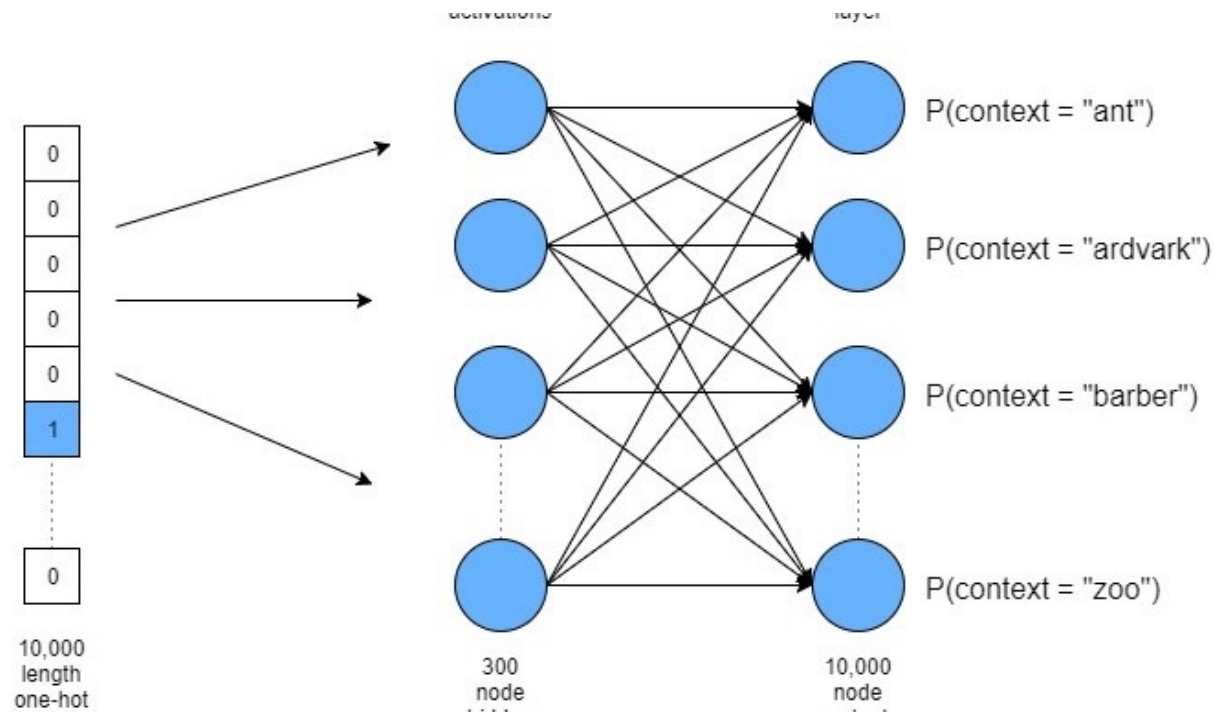
- Distancia cuadrática euclidiana entre vectores de documentos normalizados:

$$d^2(\mathbf{v1}_{norm}, \mathbf{v2}_{norm}) = \left\| \frac{\mathbf{v1}}{\|\mathbf{v1}\|} - \frac{\mathbf{v2}}{\|\mathbf{v2}\|} \right\|^2 = 2 \left[ 1 - \frac{\mathbf{v1} \cdot \mathbf{v2}}{\|\mathbf{v1}\| \|\mathbf{v2}\|} \right]$$

## Parte 2. Vector encoding

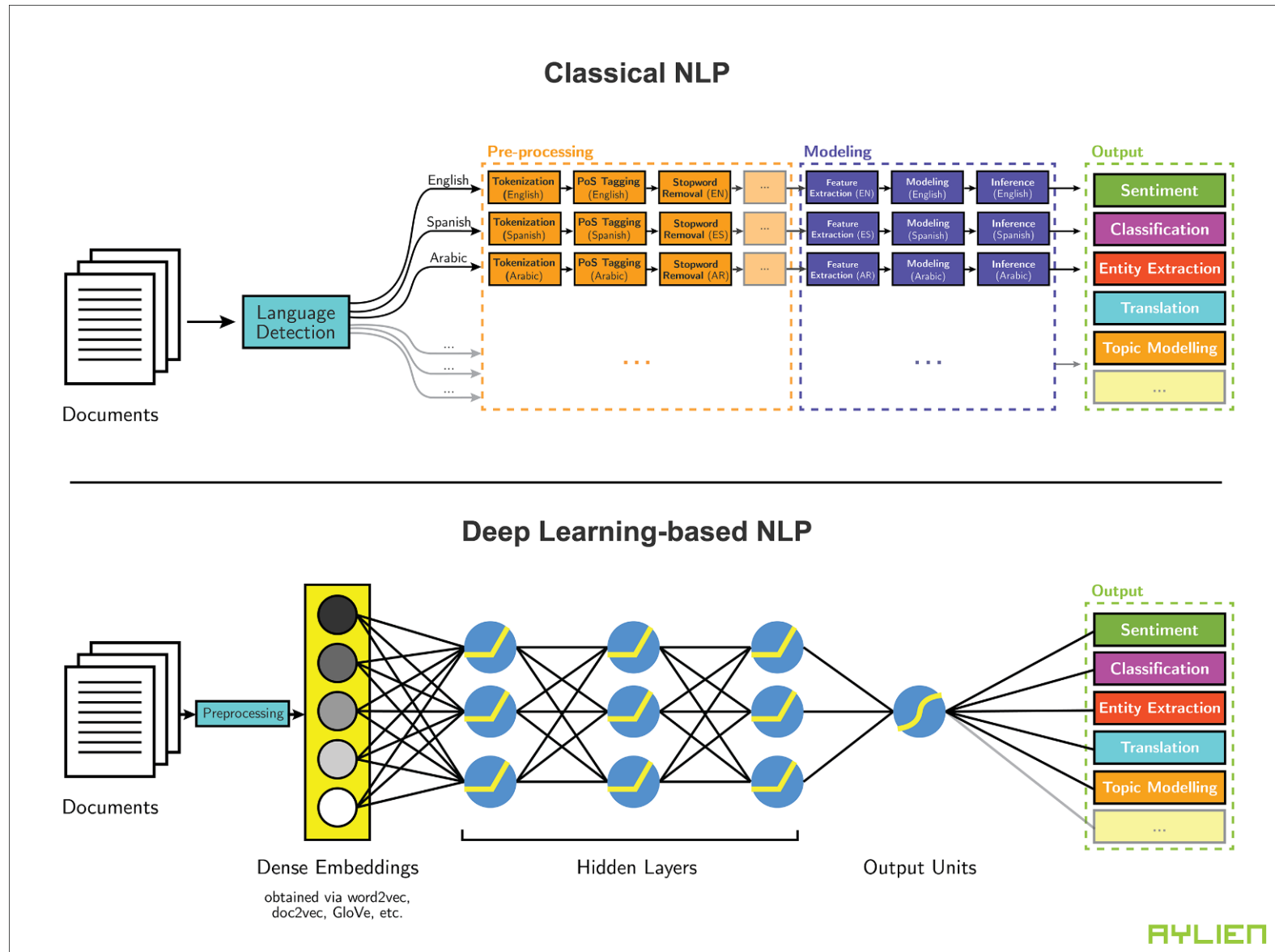
### Estrategia 4. Word2Vec

<https://radimrehurek.com/gensim/models/word2vec.html>



## Parte 2. Vector encoding

### Estrategia 5. Deep Learning Embeddings





## Parte 2. Vector encoding

---

### Estrategia 6. Transfer Learning

**Embeddings pre-entrenados con bases de datos masivas**

**GloVe:**

<https://nlp.stanford.edu/projects/glove/>

**LexVec:**

<https://github.com/alexandres/lexvec>

**FastText:**

<https://github.com/icoxfog417/fastTextJapaneseTutorial>

## Parte 3. Modelos Machine Learning

---

- **Clasificación:**

- Check first the baseline with the Dummy model
- Try first simple models as Naïve Bayes or k-Nearest Neighbours they can give good results for text classification
- Regularized Logistic Regression
- Support Vector Machines
- Ensembles (random forests, XG-Boost)

- **Clustering:**

- k-means with normalized document vectors (L2 normalized) usually gives good results

# Nuestra estrategia para procesar NL

---

1. **Preprocesado de texto**
2. **Conversión a representación Bag-of-Words**
3. **TF-IDF**
4. **Latent Semantic Analysis (LSA)**
5. **Creación de modelos descriptivos/predictivos con técnicas de Machine Learning estándar**  
Clustering, construcción de modelos predictivos, etc.

# Software

---

- Python
- NLTK library:
  - <http://www.nltk.org/>
  - <http://www.nltk.org/book/>
- Scikit-Learn, procesamiento de los textos:
  - [http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature\\_extraction.text](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text)
  - [http://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)
- Nuestras propias funciones