PREPROCESAMIENTO DE TEXTO

Sistemas inteligentes. Aplicaciones

Aránzazu Jurío Munárriz

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Codificación de un ejemplo de texto

- One-hot encoding para texto
  - *Tantas columnas como palabras existen en el diccionario*
    - Más de 93 000 palabras en el diccionario de la RAE
    - Matriz muy dispersa
    - Lento en aprendizaje y predicción

hoy feliz feliz →

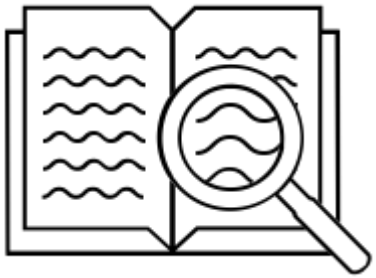
a	aba	...	feliz	...	hoy	...	zuzón
0	0	0	1	0	1	0	0

hoy feliz feliz →

a	aba	...	feliz	...	hoy	...	zuzón
0	0	0	2	0	1	0	0

Codificación de un ejemplo de texto

- One-hot encoding para texto
  - *Tantas columnas como palabras existen en el diccionario*



DB World ES | KameHameHa!!
@DB_WorldES

#BuenasNoches y #FelizFinDeSemana a todxs los fans de Dragon Ball

#BuenasNoches ✖
y
#FelizFinDeSemana ✖
a
todxs ✖

los
fans ✖
de
Dragon ✖
Ball ✖

→ y a los de

Codificación de un ejemplo de texto

- One-hot encoding para texto
  - *Utilizamos las palabras existentes en el corpus de entrenamiento*

#BuenasNoches	#FelizFinDeSemana	a	Ball	de	Dragon	fans	los	todxs	y

Codificación de un ejemplo de texto

- Separamos el texto del corpus en tokens
  - *¿Cada palabra es un token?*
  - *¿Y los signos de puntuación?*
  - *¿Y otros símbolos como # o @?*
  - *¿Hola y hola son iguales?*
  - *¿Casa y casas quiero que sean iguales o diferentes?*
  - *Palabras muy comunes como a, mi, tu, ¿necesito tenerlas en cuenta?*
  - *Si tengo muchas palabras en el corpus, ¿necesito trabajar con todas ellas? ¿Cuáles puedo eliminar?*

Preprocesamiento. Mayúsculas y minúsculas

- *¿Hola y hola son iguales?*

- Convertimos todo el texto a minúsculas (o mayúsculas)

- Dependiente del problema:
  - *granada – Granada*
  - *us – US (inglés)*

Preprocesamiento. *Tokenization*

- *¿Cada palabra es un token?*
- *¿Y los signos de puntuación?*

- Habitualmente cada palabra es un token
  - *¿Cómo tratamos los signos de puntuación (. , : ? !)*
  - *¿Y otros caracteres (# @ ☺ € $)*

Preprocesamiento. *Tokenization*

- Existen diferentes *tokenizers* que tratan estos casos especiales de diferentes maneras

Preprocesamiento. *Tokenization*

- Existen diferentes *tokenizers* que tratan estos casos especiales de diferentes maneras


```
['COMUNICADO', 'OFICIAL', '|', 'Hasbulla', 'Magomedov', 'ficha', 'por', '#Osasuna', 'hasta', '2024.', '#BienvenidoHasbulla',
'#OngiEtorriHasbulla', '#WelcomeHasbulla', '▶', 'http://bit.ly/3rga4Jd']

['COMUNICADO', 'OFICIAL', '|', 'Hasbulla', 'Magomedov', 'ficha', 'por', '#', 'Osasuna', 'hasta', '2024', '.', '#', 'Bienveni
doHasbulla', '#', 'OngiEtorriHasbulla', '#', 'WelcomeHasbulla', '▶', 'http', ':', '//bit.ly/3rga4Jd']

['COMUNICADO', 'OFICIAL', '|', 'Hasbulla', 'Magomedov', 'ficha', 'por', '#', 'Osasuna', 'hasta', '2024', '.', '#', 'Bienveni
doHasbulla', '#', 'OngiEtorriHasbulla', '#', 'WelcomeHasbulla', '▶', 'http', '://', 'bit', '.', 'ly', '/', '3rga4Jd']

['COMUNICADO', 'OFICIAL', '|', 'Hasbulla', 'Magomedov', 'ficha', 'por', '#Osasuna', 'hasta', '2024', '.', '#BienvenidoHasbul
la', '#OngiEtorriHasbulla', '#WelcomeHasbulla', '►', '', 'http://bit.ly/3rga4Jd']
```

Preprocesamiento. *Tokenization.*

- En lugar de dividir en palabras, los tokens vienen determinados por la estructura de las palabras del corpus
  - *Permite que algunas palabras de test no existentes en el corpus de train, se puedan tokenizar*
  - *Permiten subpalabras: con sentido (morfemas o arbitrarias)*
  - *Se comienza con las palabras separadas por espacio, para que caracteres de dos palabras distintas no se traten como un único token*
  - *Varios algoritmos:*
    - Byte-pair encoding
    - Unigram language modelling
    - WordPiece

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Tratamos las palabras por separado, añadiendo un símbolo de fin de palabra a cada una de ellas
- Tenemos la frecuencia de aparición de cada una de las palabras
- Inicialmente cada carácter individual es un token
- Mientras no hayamos añadido suficientes tokens:
  - *Elegimos los dos tokens contiguos se más se repitan*
  - *Creamos un nuevo token con la fusión de los dos*
  - *Añadimos ese nuevo token al vocabulario de tokens*
  - *Actualizamos el corpus con ese nuevo token*

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Corpus = [low (5 veces), lowest (2 veces), newer (6 veces), wider (3 veces), new (2 veces)

- Corpus

5 l o w _

2 l o w e s t _

6 n e w e r _

3 w i d e r _

4 n e w _

- Vocabulario

_ d e i l n o r s t w

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Corpus

5 l o w _
2 l o w e s t _
6 n e w e r _
3 w i d e r _
4 n e w _

- Vocabulario

_ d e i l n o r s t w

- Corpus

5 l o w _
2 l o w e s t _
6 n e w e r_
3 w i d e r_
4 n e w _

- Vocabulario

_ d e i l n o r s t w
r_

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Corpus

5 l o w _
2 l o w e s t _
6 n e w **e r_**
3 w i d **e r_**
4 n e w _

- Vocabulario

_ d e i l n o r s t w
r_

- Corpus

5 l o w _
2 l o w e s t _
6 n e w **er_**
3 w i d **er_**
4 n e w _

- Vocabulario

_ d e i l n o r s t w
r_ er_

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Unión
  - *(n, e)*
  - *(ne, w)*
  - *(l, o)*
  - *(lo, w)*
  - *(new, er_)*
  - *(low, _)*

- Vocabulario

(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne)

(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new)

(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo)

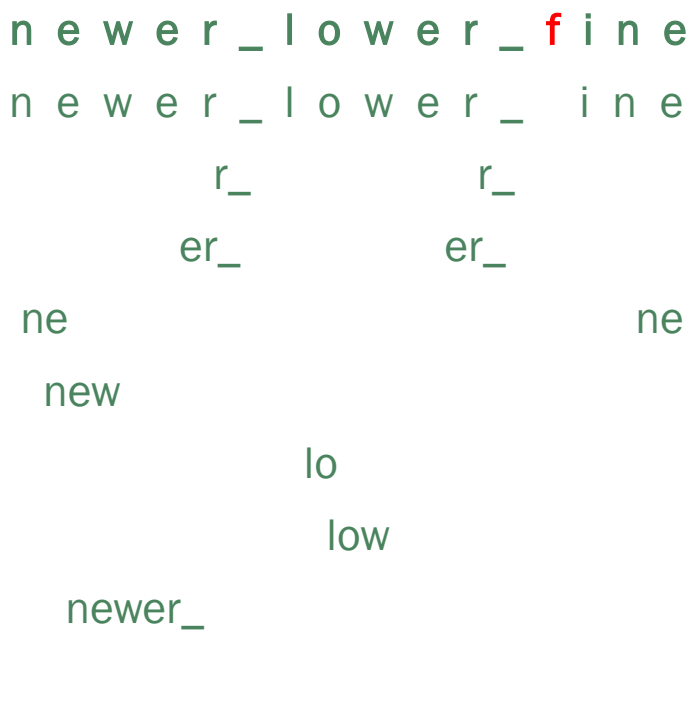
(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo, low)

(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo, low, newer_)

(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo, low, newer_, low_)

Preprocesamiento. *Tokenization. Byte-pair encoding*

- Texto de test:
  - *newer lower fine*
- Asignamos los tokens del vocabulario en orden
  - *(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo, low, newer_, low_)*



Preprocesamiento. *Tokenization. Byte-pair encoding*

- Texto de test:
  - *newer lower fine*
- Asignamos los tokens del vocabulario en orden
  - *(_, d, e, i, l, n, o, r, s, t, w, r_, er_, ne, new, lo, low, newer_, low_)*

n e w e r _ l o w e r _ f i n e _

newer_ low er_ i ne _

Preprocesamiento. Expresiones regulares

- Podemos “limpiar” el texto utilizando expresiones regulares
  - *Permiten encontrar textos que cumplan un patrón*
  - *Podemos sustituir ese patrón por el carácter (o caracteres) que nos interés, o por nada (eliminar)*

- En Python usamos el módulo re
  - *La función re.sub(patrón, nuevo_texto,texto_original) nos permite la sustitución*
  - *Para evitar problemas con las barras inversas (\) que se usan en expresiones regulares y en Python, usamos la notación de cadena raw (El string comienza con r’)*

Preprocesamiento. Expresiones regulares

- Para buscar un carácter, lo escribimos tal cual en el patrón (cuidado que las expresiones regulares son *case sensitive*)
  - *Expresión: la*
  - *Texto: La canela es la clave de las natillas*
  - *Coincidencia: La cane**la** es **la** c**la**ve de **la**s natil**la**s*
- Para expresar disyunción, utilizamos los corchetes. Podemos indicar todos los posibles caracteres dentro del corchete, o indicar un rango
  - *Expresión: [p-z]*
  - *Texto: La canela es la clave de las natillas*
  - *Coincidencia: La canela e**s** la cla**v**e de la**s** na**t**illa**s***

Preprocesamiento. Expresiones regulares

- Para indicar cualquier carácter menos los especificados, tenemos que poner como primer carácter dentro de los corchetes el circunflejo (^) y después los caracteres especificados
  - *Expresión: [^aeiou]*
  - *Texto: Nuevo^ te^xto ^de prueb^a*
  - *Coincidencia: **N**ue**v**o**^ t**e**^x**t**o **^d**e prue**b**^a* ← Los espacios también están marcados en rojo

  - *Expresión: [aeiou^]*
  - *Texto: Nuevo^ te^xto ^de prueb^a*
  - *Coincidencia: N**ue**v**o^** t**e^**x**t**o **^**d**e** pr**ue**b**^a** ← Los espacios NO están marcados en rojo

Preprocesamiento. Expresiones regulares

- Podemos indicar repeticiones de algunos caracteres dentro de la expresión regular
  - ** (asterisco): 0 o más veces*
  - *+ (más): 1 o más veces*
  - *?: 0 o 1 vez*
  - *{m}: exactamente m veces*
  - *{m,n}: entre m y n veces*
  - *Hola* → **Holaaa** que tal????*
  - *Hola+ → **Holaaa** que tal????*
  - *Hola? → **Hola**aa que tal????*
  - *Hola{2} → **Holaa**a que tal????*
  - *Hola{3,6} → **Holaaa** que tal????*

Preprocesamiento. Expresiones regulares

- Para representar cualquier carácter, utilizamos el punto (.). El único carácter que no encaja con el punto es el salto de línea
  - *Expresión: c.*ión*
  - *Texto: El **camión** lleva un cajón*
  - *Expresión: [0-9]+.[0-9]+*
  - *Texto: **123.5 123 12345** (2 match del patrón: 123.5 y 123 12345)*
  - *Expresión: [0-9]+\.[0-9]+*
  - *Texto: **123.5** 123 12345*

Preprocesamiento. Expresiones regulares

- Caracteres especiales
  - *^: además de sí mismo y negación, también significa comienzo del texto*
  - *$: final del texto*
  - *\b: comienzo o final de palabra (entendiendo como palabra secuencia de letras, dígitos y/o barras bajas*
  - *\B: no comienzo y no final de palabra*
  - *|: disyunción*
  - *dos|tres → He comprado **tres** manzanas*
  - *\bla → **la** casa de **la** colina*
  - *^la → **la** casa de la colina*

Preprocesamimento. Expresiones regulares

- Alias para conjuntos usuales
  - *\d = [0-9]*
  - *\D = [^0-9]*
  - *\w = [a-zA-Z0-9_]*
  - *\W = [^\w]*
  - *\s = [\r\t\n\f] (incluye el espacio delante del \r)*
  - *\S = [^\s]*

Preprocesamiento. Expresiones regulares

- Ejercicios:
- Sustituir todas las palabras que empiecen con ‘ca’ por ‘X’
  - *Texto: Mónica canta canciones del cancionero con su can can*
- Sustituir todos los precios menores que 1000 €, sabiendo que siempre hay un espacio entre el último dígito y el €
  - *El ordenador 1 vale 385.99 €, el ordenador 2 vale 1099.99 € y el 3 vale 750 € de oferta*
- Sustituir cualquier aparición de 500 o más GB, sabiendo que siempre hay un espacio entre el último dígito y las unidades
  - *Ese tiene 300 GB, el de ahí 1000.5 Gigabytes y el último 500 gigabyte*

Preprocesamiento. Stopwords

- Hay muchas palabras del corpus que no van a ser importantes a la hora de trabajar con él (independientemente del problema a resolver o del contexto del texto)
  - *Y, la, yo, mi, tu, su, algún, cada, con…*

- Existen muchas listas de palabras de este tipo
- Podemos eliminarlas de nuestro corpus
  - *Siempre con cuidado. Podemos perder el significado original o la estructura del texto*
    - This is not a good option → option
    - To be or not to be → null

Preprocesamiento. *Stemming y lemmatization*

- Las palabras están formadas por dos partes: raíz (lexema) y morfemas (prefijos, infijos y sufijos)
  - *La raíz tiene un significado completo*
  - *Los morfemas completan el significado de la raíz*

- Para reducir el número de palabras del corpus sin cambiar el significado (o cambiando lo mínimo posible) podemos quedarnos solo con la raíz
  - *Si nos quedamos con la palabra primitiva de la cual se deriva la raíz → lemmatization*
  - *Si nos quedamos con la raíz aunque no exista como palabra → stemming*

Preprocesamiento. *Stemming y lemmatization*

- Estoy, estás, estamos, están
  - *Lemma: estar*
  - *Stem: est*

- Sufridor, insufrible, sufriendo
  - *Lemma: sufrir*
  - *Stem: sufri*

Preprocesamiento. *Stemming*

- Es más sencillo de realizar que la lematization, por lo que es mucho más rápido.
- Para la mayoría de los problemas es suficiente
- Uno de los algoritmos más utilizamos es Porter stemmer
  - *Aplica 5 niveles de reglas para ir eliminando/modificando los sufijos de las palabras*
  - *Está diseñado para palabras en inglés*
  - *Comete errores de dos tipos:*
    - Palabras terminan en stem con diferente significado
      - *Organization – organ / numerical – numerous*
    - Palabras similares que no terminan en el mismo stem
      - *European – Europe / sparse - sparsity*

Preprocesamiento. *Porter stemmer*

- https://tartarus.org/martin/PorterStemmer/def.txt
- Medida de una palabra (m)
  - *[C](VC){m}[V] (C=consonante, V=vocal)*
- Paso 1-a (solo se aplica una regla por bloque. La primera posible)
  - *SSES → SS* *caresses → caress*
  - *IES → I* *ponies → poni*
  - *SS → SS* *caress → caress*
  - *S →* *cats → cat*
- Paso 1-b
  - *(m>0) EED → EE* *agreed → agree / feed → feed*
  - *(*v*)ED →* *plastered → plaster / bled → bled*
  - *(*v*)ING →* *motoring → motor / sing → sing*

Preprocesamiento. *Bag of words*

- Una vez que ya hemos “limpiado” el texto del corpus, ya tenemos nuestro vocabulario
- Bag of words
  - *Técnica para extraer información estructurada a partir de un texto*
  - *Suposiciones:*
    - Las palabras son independientes
    - El orden de las palabras es irrelevante

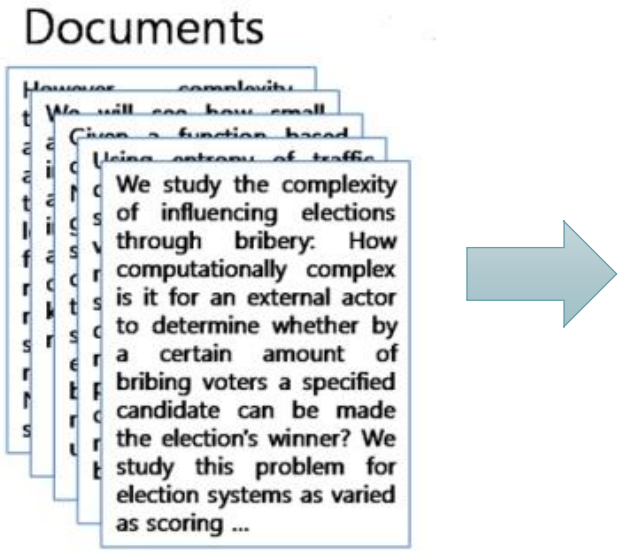
Preprocesamiento. *Bag of words*

- Cada palabra del vocabulario es una característica (o cada conjunto de palabras)
- El número de términos del corpus de entrenamiento determina la dimensión del vector
- El valor de cada elemento representa el peso (relevancia) del término en el corpus

#BuenasNoches	#FelizFinDeSemana	a	Ball	de	Dragon	fans	los	todxs	y

Preprocesamiento. *Bag of words*

- Convierte un conjunto de textos en una matriz con dimensiones: número de documentos (filas) por número de términos del vocabulario (columnas)



	complexity	algorithm	entropy	traffic	network
D1	2	3	1	0	0
D2	0	0	0	2	1
D3	3	0	0	3	4
D4	2	4	2	0	0
D5	3	4	0	0	0

Preprocesamiento. *Bag of words*

- ¿Cómo se obtienen los términos a utilizar para representar a los documentos?
  - *Todas las palabras que existen en el corpus después de su preparación → siguen siendo demasiadas*
  - *Debemos aprender los términos que mejor representan a los documentos del corpus*
- ¿Cómo se asigna el peso de cada término en cada documento?

Preprocesamiento. *Bag of words*

- ¿Son relevantes las palabras con frecuencias muy altas?
  - *Si una palabra aparece en el 95% de los documentos que tenemos, no es relevante para clasificar dichos documentos*
- ¿Son relevantes las palabras con frecuencias muy bajas?
  - *Pueden ayudarnos mucho a clasificar unos pocos textos, pero si solo aparecen en el 0.01% de los documentos, no aportan demasiada información*

Preprocesamiento. *Bag of words*

- ¿Cómo asignamos los pesos de cada término en cada documento?
  - *Pesos binarios*
  - *TF: Frecuencia de los términos (Term Frequency)*
  - *IDF: Inversa de la frecuencia en los documentos (Inverse Document Frequency)*
  - *TF-IDF: combinación de las dos anteriores*

Preprocesamiento. *Bag of words*

- Pesos binarios
  - *0 si no aparece el término en el documento*
  - *1 si aparece el término en el documento*

  - *Doc1: Text mining is to identify useful information in the text*
  - *Doc2: Useful information is mined from text*
  - *Doc3: Apple is delicious*

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
D1	1	1	1	1	0	1	1	1	0	0	0	1	1
D2	1	1	0	0	1	1	1	0	1	0	0	0	0
D3	0	0	0	0	0	1	0	0	0	1	1	0	0

Preprocesamiento. *Bag of words*

- Frecuencia de los términos
  - *Contar apariciones de cada término → los textos más largos tendrán valores mayores*
  - *Número de apariciones del término entre el total de términos del documento*
  - *Doc1: Text mining is to identify useful information in the text*
  - *Doc2: Useful information is mined from text*
  - *Doc3: Apple is delicious*

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
D1	2/10	1/10	1/10	1/10	0	1/10	1/10	1/10	0	0	0	1/10	1/10
D2	1/6	1/6	0	0	1/6	1/6	1/6	0	1/6	0	0	0	0
D3	0	0	0	0	0	1/3	0	0	0	1/3	1/3	0	0

Preprocesamiento. *Bag of words*

- Inversa de la frecuencia en los documentos
  - *Asignar pesos más grandes a los términos que no son comunes en el corpus → mayor poder de diferenciación*
- Clasificar noticias de fútbol en deportes o corrupción
  - *Futbolista: saldrá en casi todos los documentos. No ayuda a determinar el tipo de noticia → peso pequeño*
  - *Evasión: saldrá en menos documentos, pero solo en aquellos relacionados con la corrupción → mayor peso*
- El peso se calcula en base a todo el corpus, no a cada documento
  - $IDF(t) = 1 + log\left(\frac{N}{df(t)}\right)$
    - t es el término
    - N es el número de documentos del corpus
    - df(t) es el número de documentos que contienen el término t
- Si el término aparece en el documento, se le asigna el peso calculado de manera general, si no, se le asigna 0

Preprocesamiento. *Bag of words*

- Inversa de la frecuencia en los documentos
  - *Doc1: Text mining is to identify useful information in the text*
  - *Doc2: Useful information is mined from text*
  - *Doc3: Apple is delicious*

	text	information	identify	mining	mined	is	Useful	to	from	apple	delicious	in	the
N	3	3	3	3	3	3	3	3	3	3	3	3	3
df(t)	2	2	1	1	1	3	2	1	1	1	1	1	1
IDF	1.41	1.41	2.10	2.10	2.10	1	1.41	2.10	2.10	2.10	2.10	2.10	2.10

	text	information	Identify	mining	mined	is	Useful	to	From	apple	Delicious	in	the
D1	1.41	1.41	2.10	2.10	0	1	1.41	2.10	0	0	0	2.10	2.10
D2	1.41	1.41	0	0	2.10	1	1.41	0	2.10	0	0	0	0
D3	0	0	0	0	0	1	0	0	0	2.10	2.10	0	0

Preprocesamiento. *Bag of words*

- TF-IDF
  - *Valorar los términos que no son muy comunes en el corpus (IDF alto) pero que tienen un nivel de frecuencia razonable (TF alto)*
  - *Es el método más habitual para asignar los pesos*
  - *Fórmula general*
    - $TF - IDF(t) = TF(t) * IDF(t)$
  - *Fórmula utilizada comúnmente*
    - $TF - IDF(t) = TF(t) * log\left(\frac{N}{df(t)}\right)$