Información no estructurada

Textos

Manuel Sánchez-Montañés

Contenidos

- Etapas de preprocesamiento fundamentales en un proyecto de analítica de textos
- ¿Qué es un embedding?
- Estrategias de embedding más comunes: Latent Semantic Analysis (LSA), Word2Vec, capas de embedding de redes profundas, transfer learning
- Creación en Python de scripts sencillos para realizar analítica de textos usando librerías como NLTK o Scikit-Learn. Ejemplo con una base de datos real

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 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?

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 los textos

- **Lematizador**

  "cables" ⇒ "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"]

  ⇒ ["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”]

$$tf = \begin{bmatrix} \text{bought} & \text{car} & \text{cost} & \text{energy} & \text{failure} & \text{frequent} & \text{low} & \text{motorcyc} & \text{quality} \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

número de documentos

Número de palabras

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/número) o log(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”
- $idf[t] = 1 + log\left[\frac{D+1}{df[t]+1}\right]$ = [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 tf-idf 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* x *W* (D: número de documentos; W: número de palabras) puede descomponerse **exactamente** como

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

donde:

- Σ es una matriz diagonal con componentes no negativos y dimensiones LxL
- L ("dimensiones del espacio latente") =min(D,W)
- U es una matriz de *D* x *L* dimensiones donde cada columna está normalizada y es ortogonal a las otras
- V es una matriz de *W* x *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

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

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

$$V = \begin{bmatrix} -0.03996021 & 0.49070921 & -0.14609557 & -0.16990314 \\ -0.44016616 & -0.11374381 & 0.11443914 & -0.31889378 \\ -0.27669756 & 0.04041836 & 0.51231256 & -0.30320033 \\ -0.44016616 & -0.11374381 & 0.11443914 & 0.8370474 \\ -0.03996021 & 0.49070921 & -0.14609557 & 0.11095191 \\ -0.32693719 & -0.30832433 & -0.79574683 & -0.15160017 \\ -0.48012637 & 0.3769654 & -0.03165643 & -0.05200069 \\ -0.03996021 & 0.49070921 & -0.14609557 & 0.11095191 \\ -0.44016616 & -0.11374381 & 0.11443914 & -0.1629526 \end{bmatrix}$$

Parte 2. Vector encoding

Estrategia 3: Latent Semantic Analysis (LSA), 3

- Las componentes de Σ se interpretan como las "importancias" de cada dimensión latente y se ordenan en orden decreciente:

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

- 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:

$U =$
```
[[-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]]
```

$\Sigma =$
```
[[3.87493377 0.         0.         0.        ]
 [0.         1.94118656 0.         0.        ]
 [0.         0.         1.79351137 0.        ]
 [0.         0.         0.         0.        ]]
```

$V =$
```
[[-0.03996021  0.49070921 -0.14609557 -0.16990314]
 [-0.44016616 -0.11374381  0.11443914 -0.18897078]
 [-0.27669756  0.04041836  0.51231256 -0.33232033]
 [-0.44016616 -0.11374381  0.11443914  0.85770474]
 [-0.03996021  0.49070921 -0.14609557  0.11695191]
 [-0.32693719 -0.30832433 -0.79574683 -0.15790017]
 [-0.48012637  0.3769654  -0.03165643 -0.07200069]
 [-0.03996021  0.49070921 -0.14609557  0.11095191]
 [-0.44016616 -0.11374381  0.11443914 -0.16295176]]
```

(esto es, reducimos el número de columnas en las matrices)

Si ahora calculamos $U \cdot \Sigma \cdot V^T$ obtenemos de nuevo la matriz M original:

```
[[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]]
```

Parte 2. Vector encoding

Estrategia 3: Latent Semantic Analysis (LSA), 5

- Uso práctico de LSA:

  Computar la descomposición SVD de la matriz tf-idf

  Eliminar las dimensiones latentes con menos información

  Eliminar las correspondientes columnas en U, V, Σ

  Ahora la fila d en U· Σ se interpreta como la vectorización del documento d

  La fila w en V· Σ se interpreta como la vectorización de la palabra d

- 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"

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

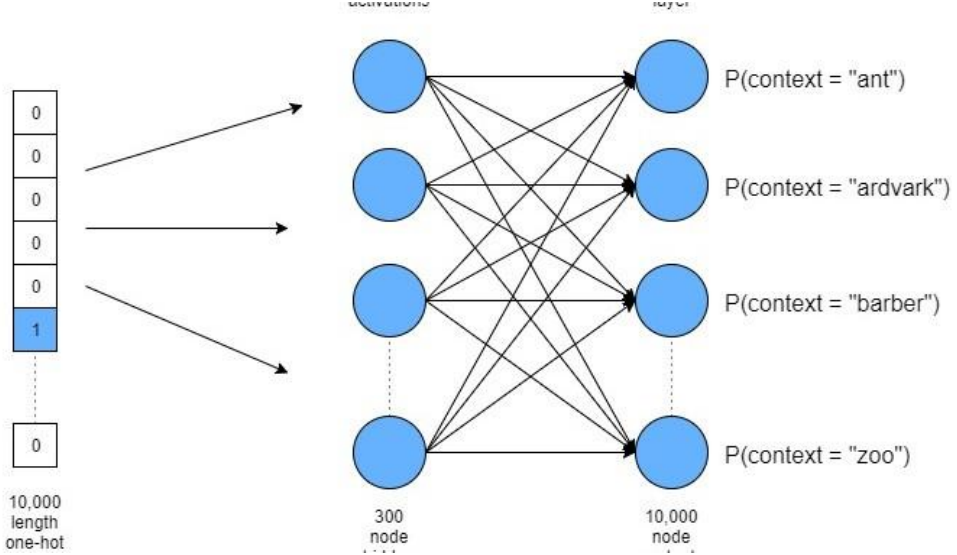
- Distancia cuadrática euclidiana entre vectores de documentos normalizados:

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

Parte 2. Vector encoding

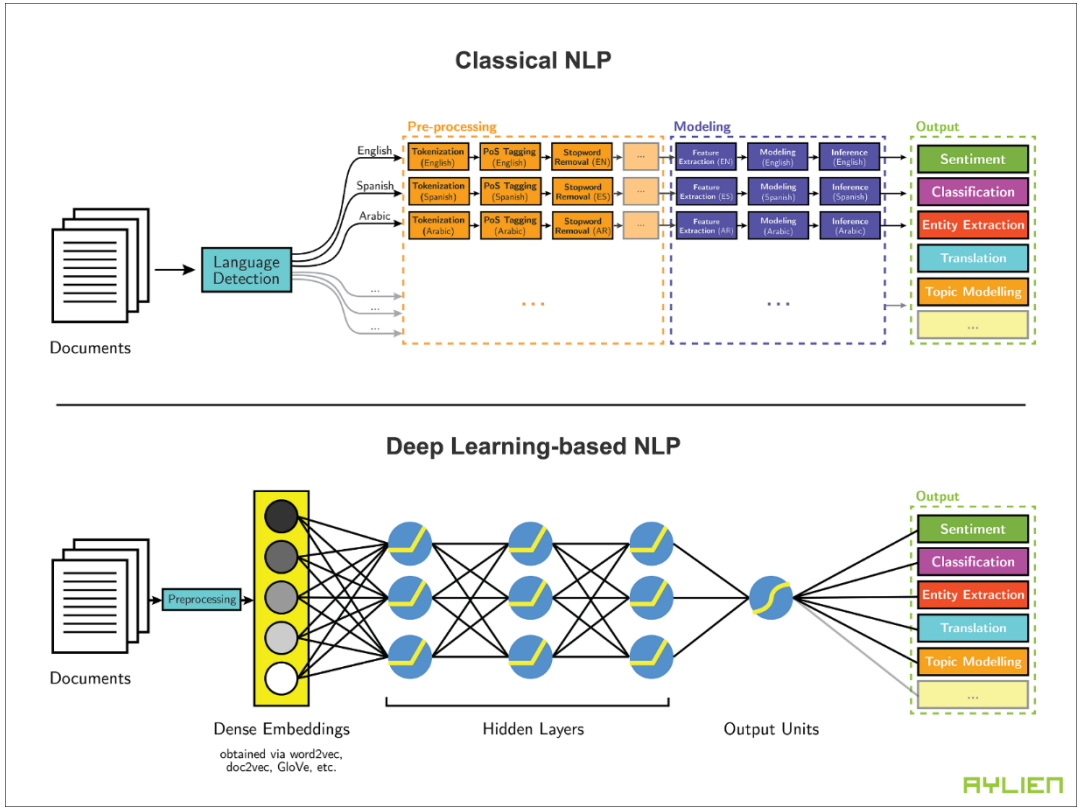
Estrategia 4. Word2Vec

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 datas masivas

GloVe:

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

LexVec:

https://github.com/alexandres/lexvec

FastText:

https://github.com/icoxfog417/fastTextJapaneseTutorial

Parte 3. Creación de modelos de Machine Learning

- **Clasificación:**
  - Chequear primero baseline con modelo Dummy (clasificación por mayoría)
  - Probar primero modelos sencillos como Naïve Bayes o k-NN ya que pueden dar buenos resultados en clasificación de textos
  - Regresión logística regularizada
  - Support Vector Machines
  - Ensembles (random forest, XG-Boost)
- **Clustering:**
  - k-means con vectores de documentos normalizados (típicamente con L2) suele dar buenos resultados

Nuestra estrategia para procesar leanguaje natural

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
- Librería NLTK:

  http://www.nltk.org/

  http://www.nltk.org/book/

- Scikit-Learn, procesado de los textos:
- 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