clusterAI 2020
ciencia de datos en ingeniería industrial
UTN BA
curso I5521

clase_09: Intro a Procesamiento del Lenguaje Natural

Docente: Martin Palazzo

agenda clase09

Preprocesamiento del Lenguaje Natural

- Tokenization
- Bag of Words (BoW)
- TF-IDF

NLP in a nutshell

Para trabajar con Lenguaje Natural (texto) tenemos que realizar una serie de pasos antes de poder entrenar un modelo. Es decir, buscar la forma de convertir el texto a una representación numérica/vectorial que pueda ser interpretable por los algoritmos de aprendizaje automático.

Tokenization

Dada una secuencia de caracteres y una unidad de documento definida, tokenization es la tarea de separar el texto en piezas mas pequeñas llamadas “tokens”.

Ejemplo: “Este fin de semana voy a hacer la tarea de cluster”

Tokens: [‘Este’, ‘fin’, ‘de’, ‘semana’, ‘voy’, ‘a’, ‘hacer’, ‘la’, ‘tarea’, ‘de’, ‘cluster’]

Bag of Words vectorizer

Es un método de pre procesamiento de lenguaje natural que toma un documento y determina la frecuencia de aparición de cada palabra (token) en el documento.

Supongan que el conjunto de documentos tiene las siguientes 3 frases:
Doc1: “Quiero hacer la asignación de Cluster”
Doc2: “Ya hice la asignación de Cluster. Que buena asignación”
Doc3: “Ya fui a UTN”

Bag of Words vectorizer

Features

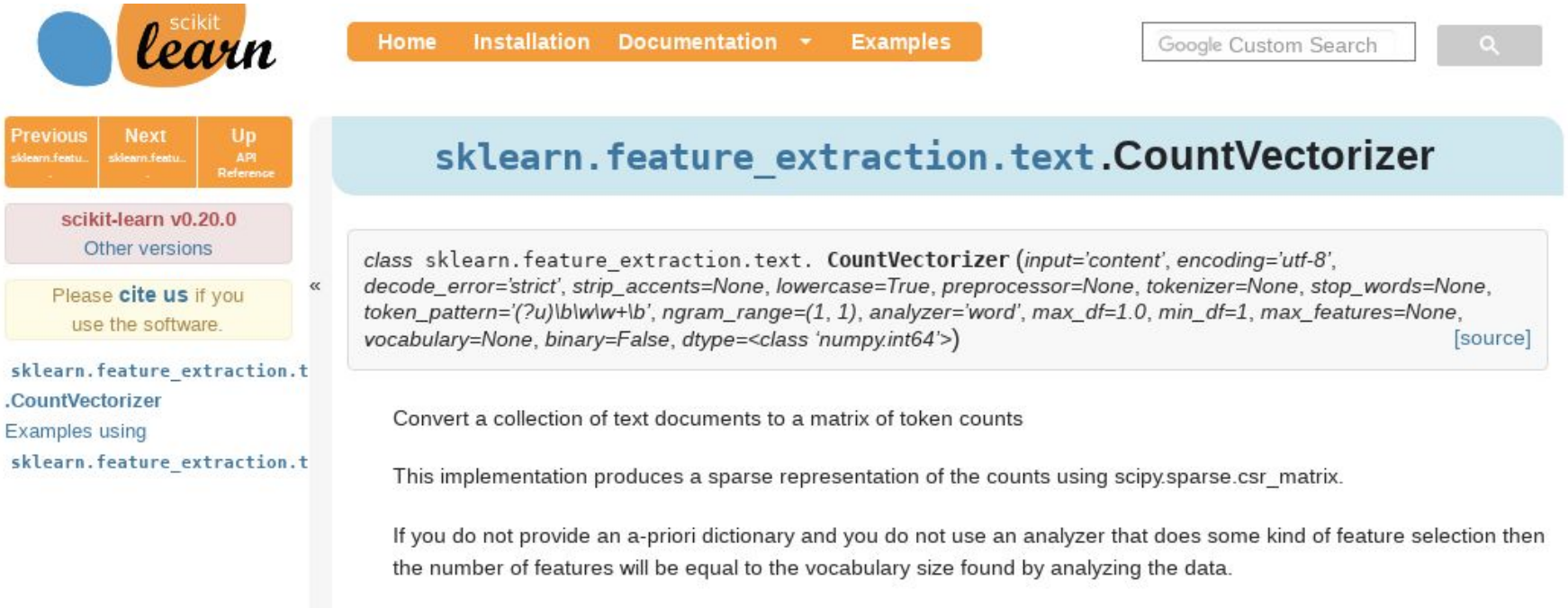
Samples

	Quiero	Hacer	la	asignacion	de	Cluster	Ya	hice	fui	a	que	buena
Doc1	1	1	1	1	1	1	0	0	0	0	0	0
Doc2	0	0	1	2	1	1	1	1	0	0	1	1
Doc3	0	0	0	0	0	0	1	0	1	1	0	0

Frecuencia del término “asignación” en el documento 2

Aplicando el Bag of Words vectorizer obtenemos una matriz de n x m lista para poder ser utilizada por modelos predictivos donde cada token es una feature y cada documento una instancia/sample.

Bag of Words = CountVectorizer (sklearn)



scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous Next Up

API Reference

scikit-learn v0.20.0

Other versions

Please cite us if you use the software.

sklearn.feature_extraction.t

.CountVectorizer

Examples using

sklearn.feature_extraction.t

sklearn.feature_extraction.text.CountVectorizer

class sklearn.feature_extraction.text. **CountVectorizer** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) [source]

Convert a collection of text documents to a matrix of token counts

This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix.

If you do not provide an a-priori dictionary and you do not use an analyzer that does some kind of feature selection then the number of features will be equal to the vocabulary size found by analyzing the data.

TF-IDF: term frequency – inverse document frequency

- **BoW** solamente considera la frecuencia total de una palabra en todos los documentos y no contempla si una palabra fue frecuente en un solo documento y no en el resto.

- **TF-IDF** contempla si una palabra con alta frecuencia aparece en un solo texto o en todos, asignando un “score” mayor en el primer caso y penalizando el segundo.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

TF: TERM FREQUENCY

N: NUMERO DE DOCUMENTOS

DF: NUM. DE DOCS. QUE CONTIENEN I

TF IDF vectorizer

Features

Samples

	Quiero	Hacer	la	asignacion	de	Cluster	Ya	hice	fui	a	que	buena
Doc1	0.2	0.3	0	0.1	0	0.4	0	0	0	0	0	0
Doc2	0	0	0.4	0.15	0.15	0.35	0.1	0.3	0	0	0.4	0.1
Doc3	0	0	0	0	0	0	0.1	0	0.3	0.5	0	0

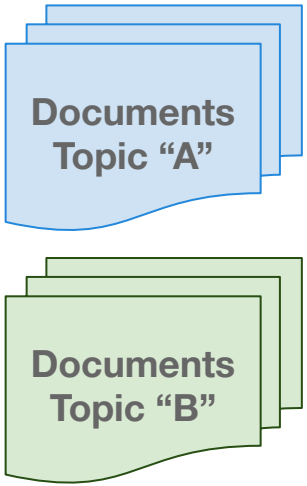
Aplicando el TF-IDF vectorizer obtenemos una matriz de n x m lista para poder ser utilizada por modelos predictivos.

Frecuencia del término “asignación” en el documento 2 * Frecuencia Inversa del término en todos los documentos

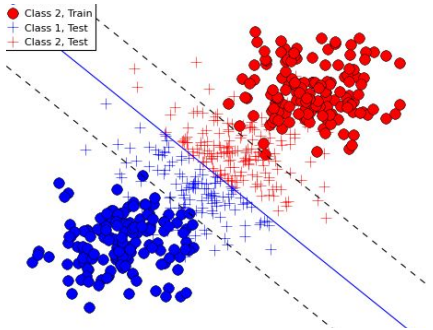
Bag of Words = CountVectorizer (sklearn)

Sentiment Analysis Data Pipeline

Tokenizing raw text → **Vectorize tokenized data (tf-idf o BoW)** → **Train model with vectorized data** → **Test model**



$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$



	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

Otras técnicas en NLP

- Stemming
- One-hot-encoding embeddings
- Word2Vec
- RNN para secuencias de texto