Información PLN

FUENTE: <https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk>

Text mining is a process of exploring sizeable textual data and find patterns. Text Mining process the text itself, while NLP process with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Text mining is one of the components of Natural language processing. NLP helps identified sentiment, finding entities in the sentence, and category of blog/article.

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms.

### Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

### Graficar por frecuencia

from nltk.probability import FreqDist

fdist = FreqDist(tokenized\_word)

fdist.most\_common(2)

import matplotlib.pyplot as plt

fdist.plot(30,cumulative=False)

plt.show()

### Stopwords

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

from nltk.corpus import stopwords

stop\_words=set(stopwords.words("english"))

print(stop\_words)

### Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

### Stemming

### Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

### from nltk.stem import PorterStemmer

### from nltk.tokenize import sent\_tokenize, word\_tokenize

### ps = PorterStemmer()

### stemmed\_words=[]

### for w in filtered\_sent:

### stemmed\_words.append(ps.stem(w))

print("Filtered Sentence:",filtered\_sent)

print("Stemmed Sentence:",stemmed\_words)

### Lemmatization

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context.

from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()

### pos tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

sent = "Albert Einstein was born in Ulm, Germany in 1879."

tokens=nltk.word\_tokenize(sent)

print(tokens)

['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']

nltk.pos\_tag(tokens)

### Sentiment Analysis

Quantifying users content, idea, belief, and opinion is known as sentiment analysis. User's online post, blogs, tweets, feedback of product helps business people to the target audience and innovate in products and services. Sentiment analysis helps in understanding people in a better and more accurate way. It is not only limited to marketing, but it can also be utilized in politics, research, and security.

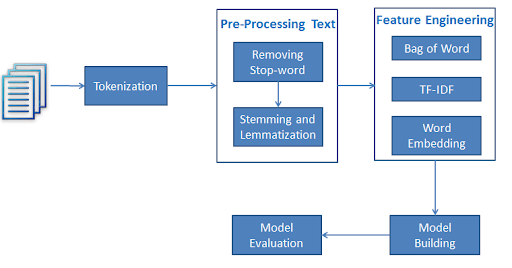
Human communication just not limited to words, it is more than words. Sentiments are combination words, tone, and writing style. As a data analyst, It is more important to understand our sentiments, what it really means?

There are mainly two approaches for performing sentiment analysis.

* Lexicon-based: count number of positive and negative words in given text and the larger count will be the sentiment of text.
* Machine learning based approach: Develop a classification model, which is trained using the pre-labeled dataset of positive, negative, and neutral.

### Text Classification

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news articles, and tweets. It has various application in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc.



### Feature Generation using Bag of Words

In the Text Classification Problem, you need to convert these text into some numbers or vectors of numbers.

Bag-of-words model(BoW ) is the simplest way of extracting features from the text. BoW converts text into the matrix of occurrence of words within a document. This model concerns about whether given words occurred or not in the document.

You can generate document term matrix by using scikit-learn's CountVectorizer.

from sklearn.feature\_extraction.text import CountVectorizer

from nltk.tokenize import RegexpTokenizer

#tokenizer to remove unwanted elements from out data like symbols and numbers

token = RegexpTokenizer(r'[a-zA-Z0-9]+')

cv = CountVectorizer(lowercase=True,stop\_words='english',ngram\_range = (1,1),tokenizer = token.tokenize)

text\_counts= cv.fit\_transform(data['Phrase'])

Luego, text\_count es el objeto utilizado para entrenar y testear el modelo. Ejemplo:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split( text\_counts, data['Sentiment'], test\_size=0.3, random\_state=1)

from sklearn.naive\_bayes import MultinomialNB

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Generation Using Multinomial Naive Bayes

clf = MultinomialNB().fit(X\_train, y\_train)

predicted= clf.predict(X\_test)

print("MultinomialNB Accuracy:",metrics.accuracy\_score(y\_test, predicted))

FUENTE:<https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/>

### Estrategias para el manejo de Datos Desbalanceados:

**Ajuste de Parámetros del modelo:** Consiste en ajustar parametros ó metricas del propio algoritmo para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento. Ejemplos on ajuste de peso en árboles, también en logisticregression tenemos el parámetro class\_weight= “balanced” que utilizaremos en este ejemplo. No todos los algoritmos tienen estas posibilidades. En redes neuronales por ejemplo podríamos ajustar la métrica de Loss para que penalice a las clases mayoritarias.

**Modificar el Dataset**: podemos eliminar muestras de la clase mayoritaria para reducirlo e intentar equilibrar la situación. Tiene como “peligroso” que podemos prescindir de muestras importantes, que brindan información y por lo tanto empeorar el modelo. Entonces para seleccionar qué muestras eliminar, deberíamos seguir algún criterio. También podríamos agregar nuevas filas con los mismos valores de las clases minoritarias, por ejemplo cuadriplicar nuestras 492 filas. Pero esto no sirve demasiado y podemos llevar al modelo a caer en overfitting.

## Subsampling en la clase mayoritaria: Lo que haremos es utilizar un algoritmo para reducir la clase mayoritaria.

* Oversampling de la clase minoritaria: En este caso, crearemos muestras nuevas “sintéticas” de la clase minoritaria.
* Combinamos resampling con Smote-Tomek: Ahora probaremos una técnica muy usada que consiste en aplicar en simultáneo un algoritmo de subsampling y otro de oversampling a la vez al dataset. En este caso usaremos SMOTE para oversampling: busca puntos vecinos cercanos y agrega puntos “en linea recta” entre ellos. Y usaremos Tomek para undersampling que quita los de distinta clase que sean nearest neighbor y deja ver mejor el decisión boundary (la zona limítrofe de nuestras clases).

**Muestras artificiales**: podemos intentar crear muestras sintéticas (no idénticas) utilizando diversos algoritmos que intentan seguir la tendencia del grupo minoritario. Según el método, podemos mejorar los resultados. Lo peligroso de crear muestras sintéticas es que podemos alterar la distribución “natural” de esa clase y confundir al modelo en su clasificación.

**Balanced Ensemble Methods**: Utiliza las ventajas de hacer ensamble de métodos, es decir, entrenar diversos modelos y entre todos obtener el resultado final (por ejemplo “votando”) pero se asegura de tomar muestras de entrenamiento equilibradas.