DNS lab
Data and Network Science Lab

UFAM

Análise de Sentimento em Redes Sociais Utilizando Scikit-learn+NLTK

Thais Gomes de Almeida

tga@icomp.ufam.edu.br

DNS lab - Pesquisadores

DNS lab
Data and Network Science Lab

Contextualização

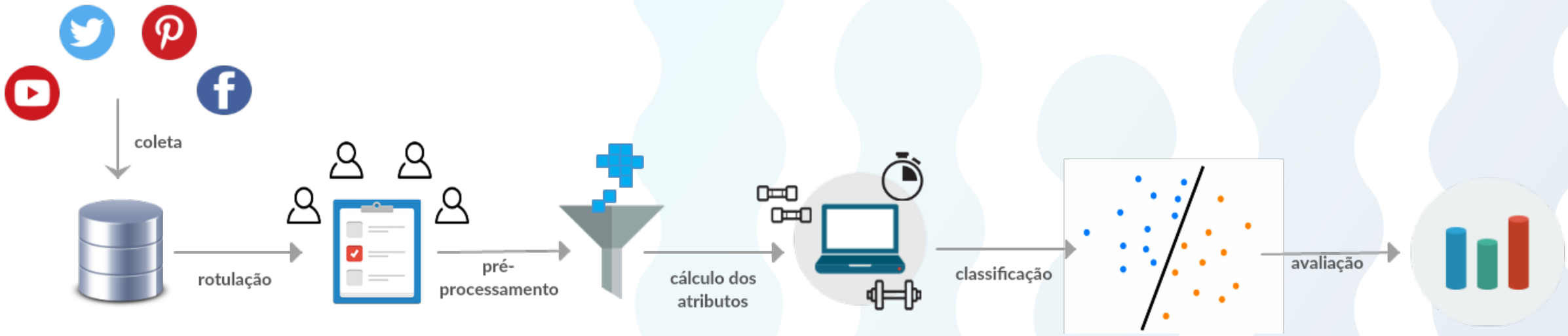
DNS lab
Data and Network Science Lab

DNS lab
Data and Network Science Lab

Aprendizagem de Máquina Supervisionada

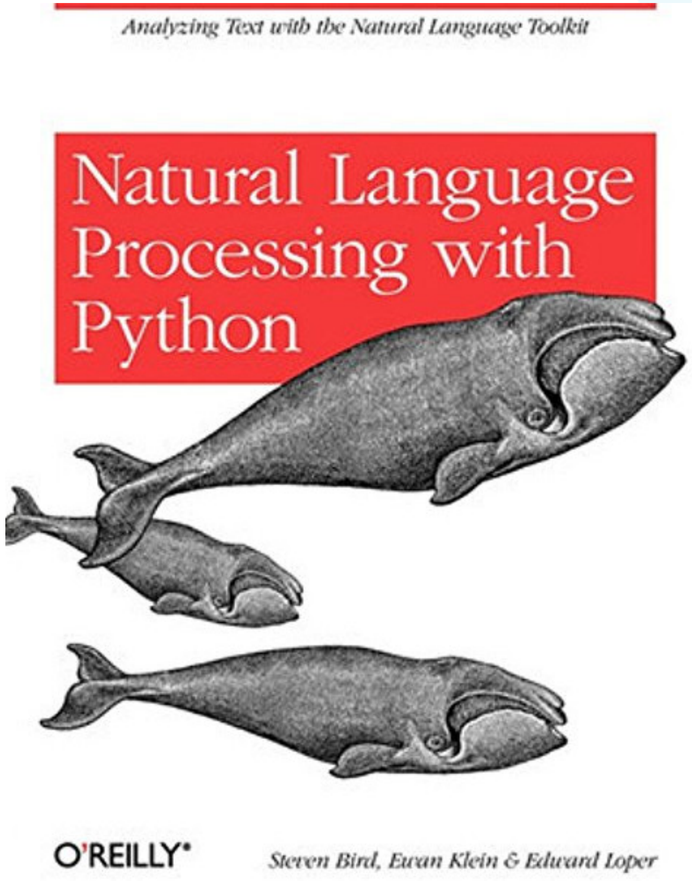
"Um conjunto de exemplos de **treinamento** com as **respostas corretas** (alvos) é fornecido e, com base neste conjunto de treinamento, o algoritmo geral consegue responder corretamente a todas as entradas possíveis." (Herbrich e Graepel, 2014)

Arquitetura Clássica de Aprendizagem Supervisionada



DNS lab
Data and Network Science Lab

Natural Language Toolkit - NLTK



- Atualmente na **versão 3.4.2**;
- Torna ágil operações como ***tokenization***, ***stemming***, ***lemmatization***;
- Possui implementações de métodos supervisionados e de Reconhecimento de Entidade Nomeda;

NLTK: Pré-Processamento

```
list_docs = []
list_label = []

with open('dataset_foursquare_tips.csv') as csvfile:
    dataset = csv.reader(csvfile, delimiter=',')
    for line in dataset:
        # pre-precess considering words
        tips_text = remove_ulr_corpus(line[0])
        tips_text = remove_mention_corpus(tips_text)
        tips_text = tokenize_words_corpus(tips_text)
        tips_text = remove_stopwords_corpus(tips_text)
        document = ' '.join(tips_text)

        list_docs.append(document)
        list_label.append(line[1])
```

NLTK: Pré-Processamento

```
list_docs = []
list_label = re.sub(r'^https?:\/\/.*[\r\n]*', '', text)

with open('dataset_foursquare_tips.csv') as csvfile:
    dataset = csv.reader(csvfile, delimiter=',')
    for line in dataset:
        # pre-precess considering words
        tips_text = remove_ulr_corpus(line[0])
        tips_text = remove_mention_corpus(tips_text)
        tips_text = tokenize_words_corpus(tips_text)
        tips_text = remove_stopwords_corpus(tips_text)
        document = ' '.join(tips_text)

        list_docs.append(document)
        list_label.append(line[1])
```

NLTK: Pré-Processamento

```
list_docs = []
list_label = []

with open('dataset_foursquare_tips.csv') as csvfile:
    dataset = csv.reader(csvfile, delimiter=',')
    for line in dataset:
        # pre-precess considering words
        tips_text = remove_ulr_corpus(line[0])
        tips_text = remove_mention_corpus(tips_text)
        tips_text = tokenize_words_corpus(tips_text)
        tips_text = remove_stopwords_corpus(tips_text)
        document = ' '.join(tips_text)

        list_docs.append(document)
        list_label.append(line[1])
```

```
re.sub(r'@\w+ ?','',text)
```

NLTK: Pré-Processamento

```python
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords

def tokenize_words_corpus(text):
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(text)
    return tokens


from nltk.stem import RSLPStemmer
from nltk.tokenize import word_tokenize

def tokenize_words_corpus(text):
    tokens = word_tokenize(text)
    porter =  RSLPStemmer()
    tokens = [porter.stem(t) for t in tokens]
    tokens = [t for t in tokens
                    if len(t)> 2 and not t.isdigit()]
    return tokens
```

NLTK: Pré-Processamento

```python
from nltk.corpus import stopwords

def remove_stopwords_corpus(tip_text):
    content = []
    for word in tip_text:
        if word.lower().strip() not in stopwords.words('portuguese'):
            content.append(word.lower().strip())
    return content

```

NLTK: Pré-Processamento

```
1  list_docs = []
2  list_label = []
3
4  with open('dataset_foursquare_tips.csv') as csvfile:
5      dataset = csv.reader(csvfile, delimiter=',')
6      for line in dataset:
7          # pre-precess considering words
8          tips_text = remove_ulr_corpus(line[0])
9          tips_text = remove_mention_corpus(tips_text)
10         tips_text = tokenize_words_corpus(tips_text)
11         tips_text = remove_stopwords_corpus(tips_text)
12         document = ' '.join(tips_text)
13
14         list_docs.append(document)
15         list_label.append(line[1])
```

Scikit-learn

- Atualmente na **versão 0.19 (julho 17)**;
- Está integrado com o **SciPy + NumPy**;
- Contém métodos de classificação, regressão, agrupamento...e muito mais!;

Scikit-learn

```
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import recall_score,precision_score,f1_score

#data representation
vectorizer = TfidfVectorizer(use_idf=True, sublinear_tf=False)
X = vectorizer.fit_transform(list_docs)
Y = list_label

#text classifier
knn = KNeighborsClassifier(n_neighbors=5)

#training
stratified_fold = StratifiedKFold(n_splits=10, shuffle=True)
predict_list = cross_val_predict(knn, X,Y, cv = stratified_fold)

#evaluate training performance
precision = precision_score(Y, predict_list, average=None)
recall = recall_score(Y, predict_list, average=None)
f1 = f1_score(Y, predict_list, average=None)
```

Scikit-learn

```
from sklearn.model_selection
from sklearn.neighbors impor
from sklearn.feature_extract
from sklearn.metrics import

#data representation
vectorizer = TfidfVectorizer(use_idf=True, sublinear_tf=False)
X = vectorizer.fit_transform(list_docs)
Y = list_label

#text classifier
knn = KNeighborsClassifier(n_neighbors=5)

#training
stratified_fold = StratifiedKFold(n_splits=10, shuffle=True)
predict_list = cross_val_predict(knn, X,Y, cv = stratified_fold)

#evaluate training performance
precision = precision_score(Y, predict_list, average=None)
recall = recall_score(Y, predict_list, average=None)
f1 = f1_score(Y, predict_list, average=None)
```

$$w_{i,j} = \begin{cases} (1 + \log f_{i,j}) \times \log \frac{N}{n_i} & \text{se } f_{i,j} > 0 \\ 0 & \text{caso contrário} \end{cases}$$

Scikit-learn

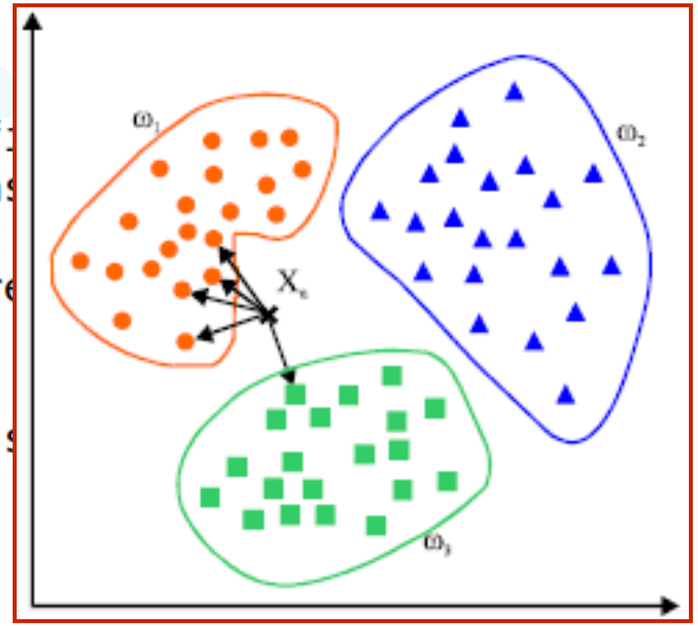
```
from sklearn.model_selection import Stratif
from sklearn.neighbors import KNeighborsClas
from sklearn.feature_extraction.text import
from sklearn.metrics import recall_score,pre

#data representation
vectorizer = TfidfVectorizer(use_idf=True, s
X = vectorizer.fit_transform(list_docs)
Y = list_label

#text classifier
knn = KNeighborsClassifier(n_neighbors=5)

#training
stratified_fold = StratifiedKFold(n_splits=10, shuffle=True)
predict_list = cross_val_predict(knn, X,Y, cv = stratified_fold)

#evaluate training performance
precision = precision_score(Y, predict_list, average=None)
recall = recall_score(Y, predict_list, average=None)
f1 = f1_score(Y, predict_list, average=None)
```



Scikit-learn

```python
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import recall_score,precision_score,f1_score

#data representation
vectorizer = TfidfVectorizer(use_idf=True, sublinear_tf=False)
X = vectorizer.fit_transform(list_docs)
Y = list_label

#text classifier
knn = KNeighborsClassifier(n_neighbors=5)

#training
stratified_fold = StratifiedKFold(n_splits=10, shuffle=True)
predict_list = cross_val_predict(knn, X,Y, cv = stratified_fold)

#evaluate training performance
precision = precision_score(Y, predict_list, average=None)
recall = recall_score(Y, predict_list, average=None)
f1 = f1_score(Y, predict_list, average=None)
```

Scikit-learn

```python
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import recall_score,precision_score,f1_score

#data representation
vectorizer = TfidfVectorizer(use_idf=True, sublinear_tf=False)
X = vectorizer.fit_transform(list_docs)
Y = list_label

#text classifier
knn = KNeighborsClassifier(n_neighbors=5)

#training
stratified_fold = StratifiedKFold(n_splits=10, shuffle=True)
predict_list = cross_val_predict(knn, X,Y, cv = stratified_fold)

#evaluate training performance
precision = precision_score(Y, predict_list, average=None)
recall = recall_score(Y, predict_list, average=None)
f1 = f1_score(Y, predict_list, average=None)
```

Considerações Finais

NumPy

matplotlib

NLTK

scikit learn

Referências Bibliográficas

- Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca. Bookman Editora, 2013.

- Marsland, Stephen. Machine learning: an algorithmic perspective. CRC press, 2015.

- Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc;

Obrigada!

Thais Gomes de Almeida

tga@icomp.ufam.edu.br