Xử lý ngôn ngữ tự nhiên

Chương 3: Phân loại văn bản

Khoa CNTT, Đại học Kỹ thuật - Công nghệ Cần Thơ

Lưu hành nội bộ

Text classification là gì?

- Quá trình hiểu và phân loại văn bản đem lại "văn bản sạch"là bước đầu tiên của xử lý ngôn ngữ.
- "Văn bản sạch"này phải đem lại ý nghĩa, kiến thức nhất định.
- Kiến thức của xử lý ngôn ngữ (knowledge of language processing) liên quan đến các khái niệm và phân tích máy học (machine learning).
- Xây dựng các hệ thống NLP.

Máy học và NLP

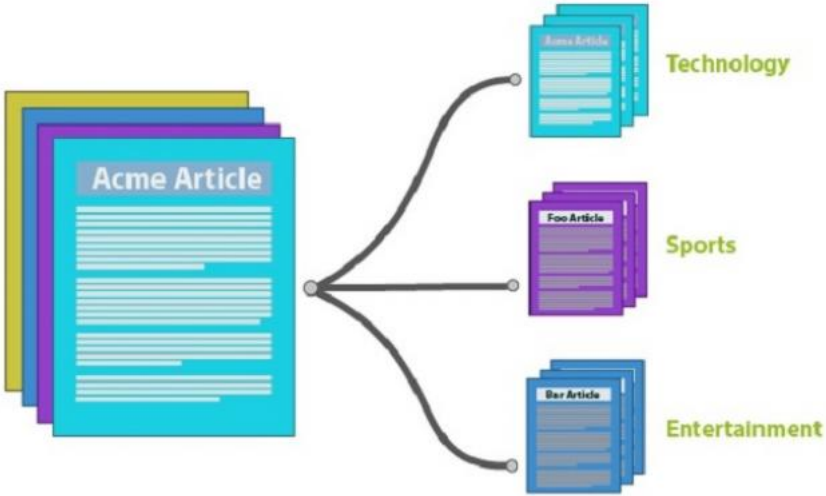
- Mục tiêu của NLP là hiểu ngôn ngữ văn bản bằng các công cụ xử lý chữ, văn phạm, từ vựng;
- Và xây dựng các hệ thống hiểu và xử lý văn bản;
- Tiến đến tự động hóa các tác vụ trên tập dữ liệu lớn.

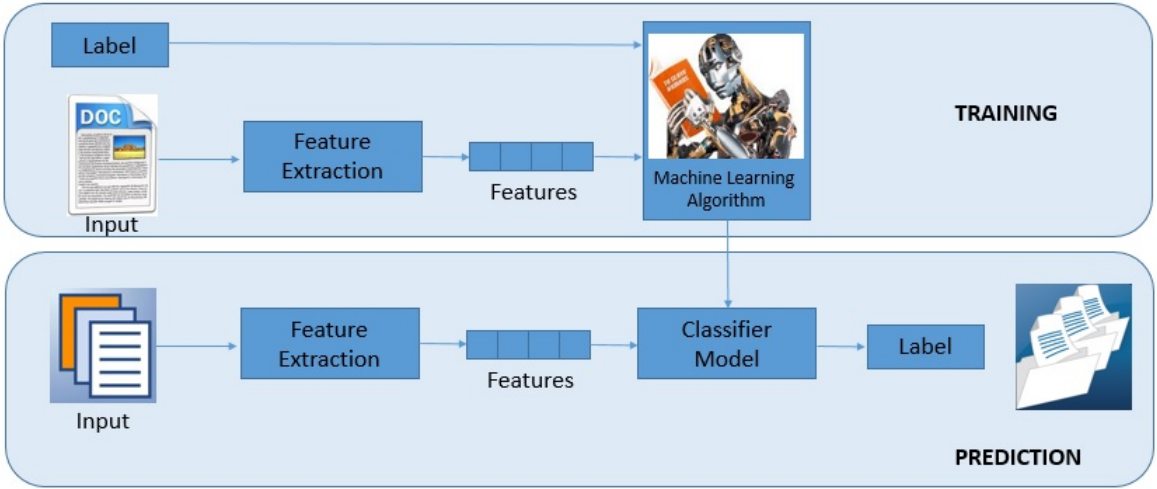
Tác vụ text classification

- Tác vụ phân loại văn bản (text classification, categorization) là một trong các tác vụ phổ biến trong NLP-ML
- Phân loại văn bản theo chủ đề dựa trên thuộc tính của từng văn bản
- Phân loại thư điện tử (spam detection)

Tác vụ text classification

- Khái niệm đơn giản của tác vụ text classification như sau:
  - Với số lượng nhỏ văn bản, người dùng đọc qua nhằm hiểu và phân thành các loại
  - Dựa trên kiến thức này, người dùng phân loại các văn bản theo loại đã xác định (document classification)
  - Làm cách nào tự động hóa với số lượng hàng triệu văn bản?
- Khái niệm classification được hiểu rộng hơn như âm nhạc, hình ảnh, phim


Acme Article
Technology
Foo Article
Sports
Bar Article
Entertainment


Label
DOC
Input
Feature Extraction
Features
Machine Learning Algorithm
TRAINING
Input
Feature Extraction
Features
Classifier Model
Label
PREDICTION

Mô hình hóa bài toán phân loại văn bản

- Dữ liệu đầu vào là văn bản (textual documents) gồm tập hợp các đoạn, câu, từ.
- Nhãn phân loại (class, category) tương ứng với mỗi dữ liệu đầu vào.
- Hệ thống phân loại học dữ liệu đầu vào và nhãn tương ứng (gia đoạn học).
- Hệ thống dự đoán các dữ liệu không có nhãn và phân loại thành các nhãn tương ứng.

Mô hình hóa bài toán phân loại văn bản

- Gọi D là tập hợp các document cần phân loại, $d \in D$ là một document bất kỳ.
- Gọi tập phân loại được định nghĩa trước là $C = \{c_1, c_2, c_3, \ldots, c_n\}$
- Gọi hệ thống phân loại văn bản $T : D \rightarrow C_i$

Tự động hóa phân loại văn bản

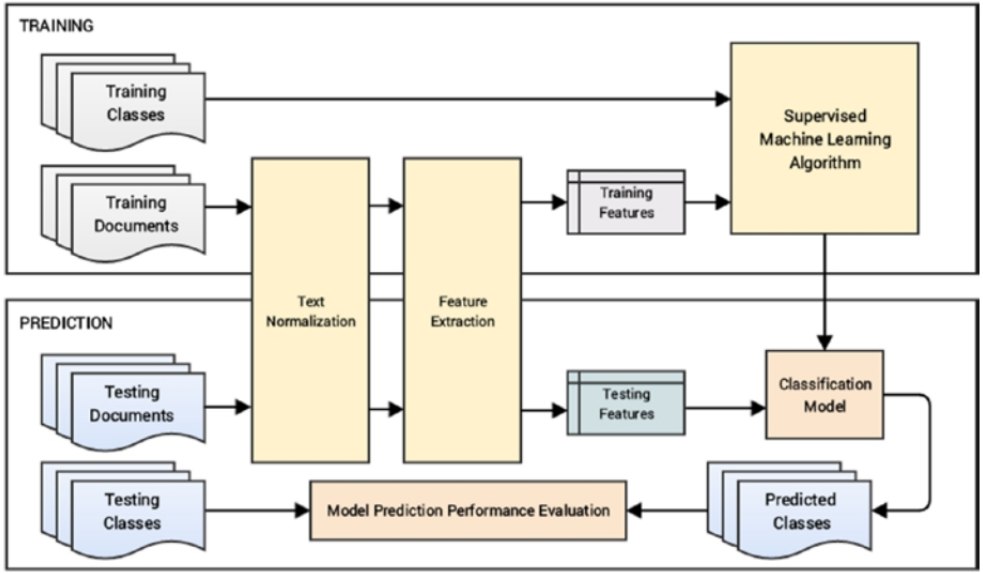
- Mặc định phần nội dung phân loại văn bản sẽ thuộc về supervised learning.
- Bài toán phân cụm văn bản (clustering) sẽ được trình bày ở chương tiếp theo.
- Hai loại bài toán của supervised learning là classification và regression.

Các tác vụ phân loại văn bản

- Binary classification: 2 nhóm classes hoặc categories
- Multi-class classification: nhiều hơn 2 nhóm classes hoặc categories
- Multi-label classification: Mỗi dự đoán tại 1 document có thể trả về nhiều classes

Các bước thực hiện

1. Prepare train and test datasets
2. Text normalization
3. Feature extraction
4. Model training
5. Model prediction and evaluation
6. Model deployment


TRAINING
Training Classes
Training Documents
Text Normalization
Feature Extraction
Training Features
Supervised Machine Learning Algorithm
PREDICTION
Testing Documents
Testing Classes
Testing Features
Classification Model
Model Prediction Performance Evaluation
Predicted Classes

Feature extraction

- Trình bày ở chương 2, không phải các token nào cũng có giá trị có ích đối với bài toán phân loại.
- feature: các thông tin về thuộc tính, tính chất của mỗi dự liệu quan sát, tạo ra sự đơn nhất có thể.
- Tác vụ liên quan đến trích lọc feature được gọi là feature extraction hoặc feature engineering.
- Features sẽ là đầu vào của bất kỳ giải thuật máy học nào.
- Máy tính chỉ có thể xử lý tính toán trên số, cho nên features phải được chuyển thành dạng số.

Vector Space Model - mô hình không gian véc-tơ

- Khái niệm Vector Space Model (Term Vector Model) là sự mô hình hóa dữ liệu văn bản sang dạng số thể hiện qua các vector tương ứng.
- Một document $d \in D$ được thể hiện bằng một vector.
- Chiều dài vector bằng tổng số token riêng biệt có trong D.
- $d = \{w_1, w_2, \ldots, w_n\}$, với w_i là các token riêng biệt trong d, n là tổng số token riêng biệt trong D.

Mô hình Bag of Words

- Sử dụng dữ liệu CORPUS và new_doc để mô tả bài toán

```
CORPUS = [
'the sky is blue',
'sky is blue and sky is beautiful',
'the beautiful sky is so blue',
'i love blue cheese'
]

new_doc = ['loving this blue sky today']
```

```
9  def bow_extractor(corpus, ngram_range=(1,1)):
10     from sklearn.feature_extraction.text import CountVectorizer
11
12     vectorizer = CountVectorizer(min_df=1, ngram_range=ngram_range)
13     features = vectorizer.fit_transform(corpus)
14
15     return vectorizer, features
16
17 bow_vectorizer, bow_features = bow_extractor(CORPUS)
18 features = bow_features.todense()
19 print (features)
20
21 #[[0  0  1  0  1  0  1  0  1]
22 #[1  1  1  0  2  0  2  0  0]
23 #[0  1  1  0  1  0  1  1  1]
24 #[0  0  1  1  0  1  0  0  0]]
```

```
feature_names = bow_vectorizer.get_feature_names()
print(feature_names)

#['and', 'beautiful', 'blue', 'cheese', 'is', 'love', 'sky', 'so', 'the
    ']

def display_features(features, feature_names):
    import pandas as pd
    df = pd.DataFrame(data=features,columns=feature_names)
    print (df)

display_features(features, feature_names)

#    and   beautiful   blue   cheese   is   love   sky   so   the
#0     0           0      1        0    1      0     1    0     1
#1     1           1      1        0    2      0     2    0     0
#2     0           1      1        0    1      0     1    1     1
#3     0           0      1        1    0      1     0    0     0
```

- Chúng ta vừa xây dựng xong 1 mô hình chuyển đổi từ dạng thể hiện chữ sang dạng thể hiện số
- Sử dụng mô hình này áp dụng cho document mới.

```
new_doc_features = bow_vectorizer.transform(new_doc)
new_doc_features = new_doc_features.todense()
print(new_doc_features)

#[[0 0 1 0 0 0 1 0 0]]

new_doc_2 = ['loving this white sky today']
new_doc_2_features = bow_vectorizer.transform(new_doc_2)
new_doc_2_features = new_doc_2_features.todense()
print(new_doc_2_features)

#[[0 0 0 0 0 0 1 0 0]]
```

```
54 display_features(new_doc_features, feature_names)
55
56 #    and   beautiful   blue   cheese   is   love   sky   so   the
57 #0     0           0      1        0    0      0     1    0      0
58
59 display_features(new_doc_2_features, feature_names)
60
61 #    and   beautiful   blue   cheese   is   love   sky   so   the
62 #0     0           0      0        0    0      0     1    0      0
```

Mô hình Bag of Words

- Mô hình Bag of Words chỉ quan tâm đến tần số xuất hiện của tokens.
- Chưa tính đến tầm quan trọng của tokens đối với document.
  - Nếu tokens chỉ xuất hiện ở một hoặc một ít documents thì có thể nói tokens đó có tầm quan trọng giúp nhận diện document tương ứng.
  - Ngược lại, nếu tokens xuất hiện ở quá nhiều documents thì không có tầm quan trọng nhận diện document tương ứng.
- $\rightarrow$ mô hình TF-IDF

Mô hình TF-IDF

- TF-IDF: Term frequency - Inverse Document Frequency

$$TFIDF = tf * idf\,, \tag{1}$$

trong đó tf biểu diễn tầm quan trọng của token với document, và idf biểu diễn tầm quan trọng của token với toàn bộ các documents.

Mô hình TF-IDF

$$\mathrm{TF}(t, d) = \frac{n^d(t)}{|d|}\ , \tag{2}$$

trong đó $n^d(t)$ là số lần token t xuất hiện trong document d và $|d|$ là tổng số tokens trong document d.

$$\mathrm{IDF}(t) = \log \frac{|C|}{n^C(t)}\ , \tag{3}$$

trong đó $|C|$ là tổng số documents đang có và $n^C(t)$ tổng số documents có chứa token t.

Mô hình TF-IDF

```
CORPUS = [
'the sky is blue',
'sky is blue and sky is beautiful',
'the beautiful sky is so blue',
'i love blue cheese'
]

new_doc = ['loving this blue sky today']

def bow_extractor(corpus, ngram_range=(1,1)):
    from sklearn.feature_extraction.text import CountVectorizer

    vectorizer = CountVectorizer(min_df=1, ngram_range=ngram_range)
    features = vectorizer.fit_transform(corpus)

    return vectorizer, features
```

```
17 def tfidf_transformer(bow_matrix):
18     from sklearn.feature_extraction.text import TfidfTransformer
19
20     transformer = TfidfTransformer(norm='l2',smooth_idf=True, use_idf=
       True)
21     tfidf_matrix = transformer.fit_transform(bow_matrix)
22     return transformer, tfidf_matrix
23
24 bow_vectorizer, bow_features = bow_extractor(CORPUS)
25 features = bow_features.todense()
26 feature_names = bow_vectorizer.get_feature_names()
```

```
import numpy as np
tfidf_trans, tdidf_features = tfidf_transformer(bow_features)
features = np.round(tdidf_features.todense(), 2)
display_features(features, feature_names)

#     and   beautiful   blue   cheese     is   love    sky     so    the
#0   0.00        0.00   0.40     0.00   0.49   0.00   0.49   0.00   0.60
#1   0.44        0.35   0.23     0.00   0.56   0.00   0.56   0.00   0.00
#2   0.00        0.43   0.29     0.00   0.35   0.00   0.35   0.55   0.43
#3   0.00        0.00   0.35     0.66   0.00   0.66   0.00   0.00   0.00
```

- Chúng ta vừa xây dựng xong 1 mô hình chuyển đổi từ dạng thể hiện bag of words sang dạng thể hiện TFIDF
- Sử dụng mô hình này áp dụng cho document mới.

```
new_doc_features = bow_vectorizer.transform(new_doc)
new_doc_tfidf = tfidf_trans.transform(new_doc_features)
new_doc_features = np.round(new_doc_tfidf.todense(), 2)
display_features(nd_features, feature_names)

#    and   beautiful   blue   cheese    is   love    sky    so   the
#0   0.0         0.0   0.63      0.0   0.0    0.0   0.77   0.0   0.0
```

```
new_doc_2 = ['loving this white sky today']

new_doc_2_features = bow_vectorizer.transform(new_doc_2)
new_doc_2_tfidf = tfidf_trans.transform(new_doc_2_features)
new_doc_2_features = np.round(new_doc_2_tfidf.todense(), 2)
display_features(new_doc_2_features, feature_names)

#    and   beautiful   blue   cheese    is   love   sky    so   the
#0   0.0         0.0    0.0      0.0   0.0    0.0   1.0   0.0   0.0
```

Mô hình dựa trên frequency

- Mô hình Bag of Words hoặc mô hình TFIDF:
  - Đều có thông tin về frequency.
  - Đều dựa trên không gian vector, số chiều vector = tổng số tokens khác nhau.
  - Cần không gian lưu trữ lớn.
- 1 biến kiểu float trong python chiếm 16 bytes bộ nhớ, nếu dữ liệu có 1000000 documents và 1000 tokens khác nhau, cần bao nhiêu GB RAM để lưu trữ dữ liệu trên?

Giải thuật phân loại

- Classification algorithms là các giải thuật thuộc nhóm supervised learning được dùng để phân loại dữ liệu thành các classes hoặc categories được định nghĩa trước.

Đánh giá mô hình phân loại

- Để đánh giá một hệ máy học hoạt động có hiệu quả hay không, ta dùng các mô hình đánh giá.
- Confusion matrix

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

```
from sklearn import metrics
import numpy as np
import pandas as pd
from collections import Counter

actual_labels = ['spam', 'ham', 'spam', 'spam', 'spam','ham', 'ham', '
    spam', 'ham', 'spam','spam', 'ham', 'ham', 'ham', 'spam','ham', '
    ham', 'spam', 'spam', 'ham']

predicted_labels = ['spam', 'spam', 'spam', 'ham', 'spam','spam', 'ham'
    , 'ham', 'spam', 'spam','ham', 'ham', 'spam', 'ham', 'ham','ham', '
    spam', 'ham', 'spam', 'spam']

ac = Counter(actual_labels)
pc = Counter(predicted_labels)

print ('Actual counts: ' +str( ac.most_common()))
print ('Predicted counts: ' + str(pc.most_common()))
```

```
#Actual counts: [('spam', 10), ('ham', 10)]
#Predicted counts: [('spam', 11), ('ham', 9)]

cm = metrics.confusion_matrix(y_true=actual_labels, y_pred=
    predicted_labels, labels=['spam','ham'])

cm_table = pd.DataFrame(data=cm,
       columns=pd.MultiIndex(levels=[['Predicted:'],
       ['spam','ham']],
       labels=[[0,0],[0,1]]),
       index=pd.MultiIndex(levels=[['Actual:'],
       ['spam','ham']],
       labels=[[0,0],[0,1]]))

print (cm_table)
```

```
#                Predicted:
#                        spam  ham
#Actual: spam             5     5
#        ham              6     4

true_positive = 5.
false_positive = 6.
false_negative = 5.
true_negative = 4.
```

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{4}$$

```
accuracy = np.round(metrics.accuracy_score(y_true=actual_labels,y_pred=
    predicted_labels),2)

accuracy_manual = np.round((true_positive + true_negative) / (
    true_positive + true_negative + false_negative + false_positive),2)

print ('Accuracy: ' + str(accuracy))
print ('Manually computed accuracy: ' + str(accuracy_manual))

#Accuracy: 0.45
#Manually computed accuracy: 0.45
```

Xây dựng một hệ thống phân loại văn bản đơn giản

- Sử dụng data 20newsgroups được cung cấp sẵn trong thư viện sk-learn
- Sử dụng kiến thức chương 2 và chương 3.
- Có 3 files code:
  - `normalization.py`: liên quan đến các tác vụ chương 2
  - `feature_extractors.py`: liên quan đến các tác vụ chương 3
  - `main_program.py`: file chương trình chính

main_program.py

```
1  from sklearn.datasets import fetch_20newsgroups
2  from sklearn.cross_validation import train_test_split
3
4  def get_data():
5      data = fetch_20newsgroups(subset='all',shuffle=True,remove=('headers
       ', 'footers', 'quotes'))
6      return data
7
8  def prepare_datasets(corpus, labels):
9      train_X, test_X, train_Y, test_Y = train_test_split(corpus, labels,
       test_size=0.33, random_state=42)
10     return train_X, test_X, train_Y, test_Y
```

```
11
12 def remove_empty_docs(corpus, labels):
13     filtered_corpus = []
14     filtered_labels = []
15     for doc, label in zip(corpus, labels):
16         if doc.strip():
17             filtered_corpus.append(doc)
18             filtered_labels.append(label)
19
20     return filtered_corpus, filtered_labels
21
22 dataset = get_data()
23
24 print (dataset.target_names)
25
26 corpus, labels = dataset.data, dataset.target
27 corpus, labels = remove_empty_docs(corpus, labels)
```

```
28
29 train_corpus, test_corpus, train_labels, test_labels = prepare_datasets
       (corpus, labels)
30
31 from normalization import normalize_corpus # file normalization.py
32
33 norm_train_corpus = normalize_corpus(train_corpus)
34 norm_test_corpus = normalize_corpus(test_corpus)
35
36 from feature_extractors import bow_extractor, tfidf_extractor # file
       feature_extractors.py
37 import nltk
```

```
38
39 # bag of words features
40 bow_vectorizer, bow_train_features = bow_extractor(norm_train_corpus)
41 bow_test_features = bow_vectorizer.transform(norm_test_corpus)
42
43 # tfidf features
44 tfidf_vectorizer, tfidf_train_features = tfidf_extractor(
       norm_train_corpus)
45 tfidf_test_features = tfidf_vectorizer.transform(norm_test_corpus)
46
47
48 # tokenize documents
49 tokenized_train = [nltk.word_tokenize(text) for text in
       norm_train_corpus]
50 tokenized_test = [nltk.word_tokenize(text) for text in norm_test_corpus
       ]
```

```
51
52 from sklearn import metrics
53 import numpy as np
54
55 def get_metrics(true_labels, predicted_labels):
56     print ('Accuracy: ' + str(np.round(metrics.accuracy_score(
       true_labels, predicted_labels),2)))
57
58
59 def train_predict_evaluate_model(classifier, train_features,
       train_labels, test_features, test_labels):
60     # build model
61     classifier.fit(train_features, train_labels)
62     # predict using model
63     predictions = classifier.predict(test_features)
64     # evaluate model prediction performance
65     get_metrics(true_labels=test_labels, predicted_labels=predictions)
66     return predictions
```

```
67
68 from sklearn.naive_bayes import MultinomialNB
69 from sklearn.linear_model import SGDClassifier
70
71 # student should try to use other algorithms
72 mnb = MultinomialNB()
73
74 # Multinomial Naive Bayes with bag of words features
75 mnb_bow_predictions = train_predict_evaluate_model(classifier=mnb,
       train_features=bow_train_features, train_labels=train_labels,
       test_features=bow_test_features, test_labels=test_labels)
76
77 # Multinomial Naive Bayes with tfidf features
78 mnb_tfidf_predictions = train_predict_evaluate_model(classifier=mnb,
       train_features=tfidf_train_features, train_labels=train_labels,
       test_features=tfidf_test_features, test_labels=test_labels)
```

```
79
80 #['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.
       ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x', 'misc.
       forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', '
       rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.
       space', 'soc.religion.christian', 'talk.politics.guns', 'talk.
       politics.mideast', 'talk.politics.misc', 'talk.religion.misc']
81 #Accuracy: 0.67
82 #Accuracy: 0.72
```