CSE: **Faculty of Computer Science and Engineering**
Thuyloi University

PHÂN LOẠI VĂN BẢN
(Text classification)

TS. Nguyễn Thị Kim Ngân
Email: ngannguyen@tlu.edu.vn

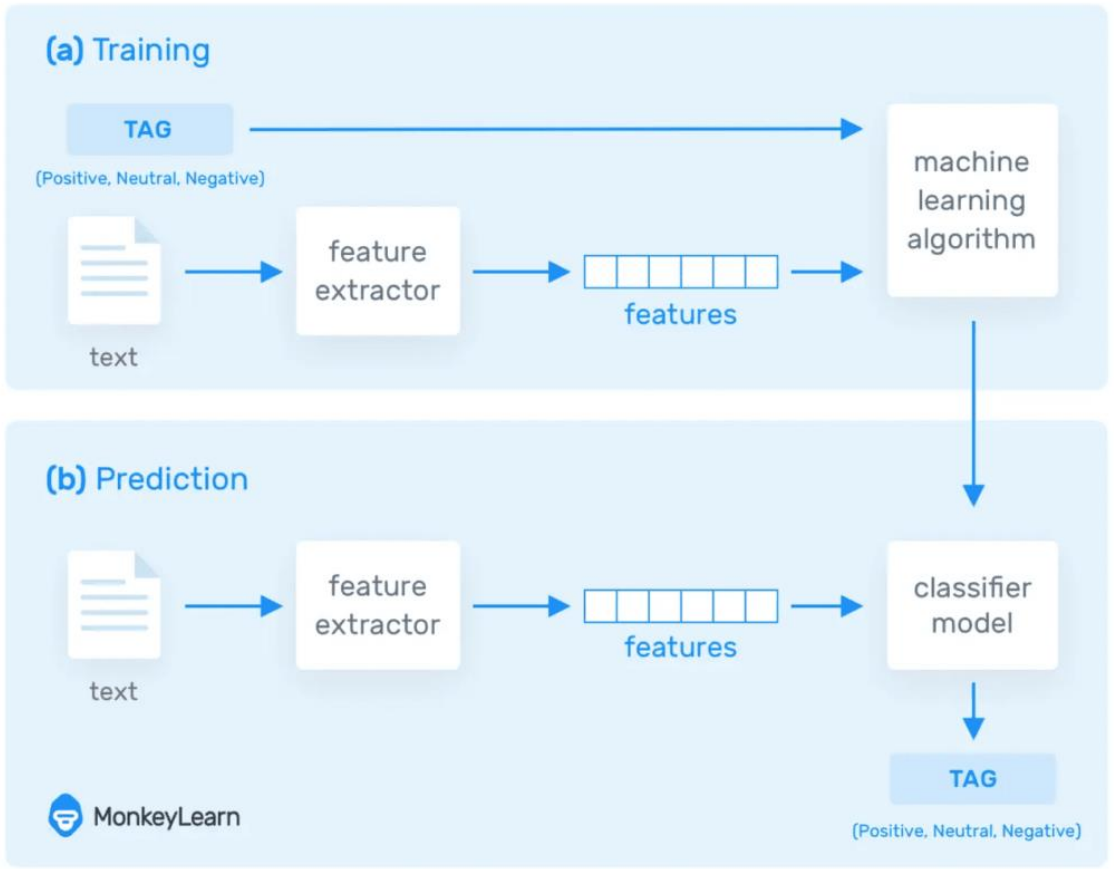
Có tham khảo một số bài viết trên https://www.miai.vn/ và https://nguyenvanhieu.vn/

Phân loại văn bản

- Là bài toán thuộc nhóm học có giám sát (Supervised learning) trong học máy. Bài toán này yêu cầu dữ liệu cần có nhãn (label)
  - Mô hình sẽ học từ dữ liệu có nhãn đó
  - Sau đó, mô hình được dùng để dự đoán nhãn cho các dữ liệu mới mà mô hình chưa gặp.

- Ví dụ: xây dựng một mô hình học máy để dự đoán chủ đề (Kinh tế, Xã hội, Thể thao,…) của một bài báo bất kỳ

Phân loại văn bản



Các bước xây dựng mô hình phân loại văn bản

- Chuẩn bị dữ liệu
- Tiền xử lý dữ liệu văn bản
- Xây dựng mô hình phân loại văn bản
- Đánh giá mô hình phân loại văn bản

Chuẩn bị dữ liệu

- Với phân loại văn bản tiếng Việt, dữ liệu cần chuẩn bị là:
- Dữ liệu các bài báo tiếng Việt và chủ đề của bài báo đó
- Có thể thu thập dữ liệu từ các website

Tiền xử lý dữ liệu văn bản

- Chuẩn hóa dữ liệu
- Loại bỏ các thành phần không có ý nghĩa

Tiền xử lý dữ liệu tiếng Việt

- Xóa HTML code (nếu có)
- Chuẩn hóa bảng mã Unicode (đưa về Unicode tổ hợp dựng sẵn)
- Chuẩn hóa kiểu gõ dấu tiếng Việt (dùng òa úy thay cho oà uý)
- Thực hiện tách từ tiếng Việt (sử dụng thư viện tách từ như pyvi, underthesea, vncorenlp,…)
- đưa về văn bản lower (viết thường)
- Xóa các ký tự đặc biệt: “.”, “,”, “;”, “)”, …

Xóa HTML code

Sử dụng regex trong Python

```
def remove_html(txt):
    return re.sub(r'<[^>]*>', '', txt)

txt = "<p class="par">This is an example</p>"
remove_html(txt)
>>> 'This is an example'
```

Chuẩn hóa Unicode tiếng Việt

- Có 2 loại mã Unicode phổ biến:
  - Unicode tổ hợp
  - Unicode dựng sẵn
- Đưa về 1 chuẩn Unicoe dựng sẵn (phổ biến hơn)

Chuẩn hóa kiểu gõ dấu

- òa với oà lần lượt là kiểu gõ cũ (phổ biến hơn) và kiểu gõ mới

Tách từ tiếng Việt

- Tách từ (Tokenize): bài toán cơ sở trong NLP
- Có một số mã nguồn mở cho bài toán tách từ tiếng Việt: underthesea, pyvi hoặc PhoBert, …

Tách từ tiếng Việt

```
from underthesea import word_tokenize
>>> sentence = 'Chàng trai 9X Quảng Trị khởi nghiệp từ nấm sò'

>>> word_tokenize(sentence)
['Chàng trai', '9X', 'Quảng Trị', 'khởi nghiệp', 'từ', 'nấm', 'sò']

>>> word_tokenize(sentence, format="text")
'Chàng_trai 9X Quảng_Trị khởi_nghiệp từ nấm sò'
```

Đưa về chữ viết thường (lowercase)

- Đưa dữ liệu về chữ viết thường là cần thiết:
  - Việc phân biệt chữ hoa, chữ thuwognf không có tác dựng ở bài toán phân loại văn bản
  - Đưa về chữ viết thường giúp giảm số lượng đặc trưng (vì máy tính hiểu hoa thường là 2 từ khác nhau) và tăng độ chính xác hơn cho mô hình

Xóa các ký tự không cần thiết

- Loại bỏ các dữ liệu không có tác dụng cho việc phân loại văn bản.
  - Giảm số chiều đặc trưng, tăng tốc độ học và xử lý
  - Tránh làm ảnh hưởng xấu tới kết quả của mô hình
- Kí tự không cần thiết:
  - Dấu ngắt câu
  - Số đếm, ngày tháng, email, …
  - Ký tự đặc biệt không giúp ích cho phân loại văn bản

Loại bỏ các stopword tiếng Việt

- Stopword là các từ xuất hiện nhiều ở tất cả các chuyên mục cần phân loại. Do đó, chúng là các đặc trưng không có tác dùng cho việc phân loại văn bản
- Các stopword thường là các từ nối (của, là, có, được, những,…) và các từ đặc trưng của dữ liệu (ví dụ như các từ "máy bay, tiếp viên" là các stopword nếu làm bài phân loại đánh giá khách hàng của doanh nghiệp vận tải hàng không

Làm sao để xây dựng bộ stopword

- Danh sách stopword nên được xây dựng từ bộ dữ liệu văn bản lớn, tầm cỡ vài chục GB là hợp lý. Nếu là dữ liệu trong miền bài toán bạn đang làm thì càng tốt!
- Xây dựng bộ từ điển stopword tiếng Việt bằng cách: thống kê các từ xuất hiện nhiều trong tất cả các chuyên mục và lấy top đầu
- Ví dụ, một số stopword:
  - và 14255
  - của 13177
  - là 9983
  - có 9162
  - ...

Loại stopword khỏi dữ liệu

- Xóa các từ trong văn bản mà có trong danh sách stopword

```
# Danh sách stopword
stopword = set()

def remove_stopwords(line):
    words = []
    for word in line.strip().split():
        if word not in stopword:
            words.append(word)
    return ' '.join(words)
```

Xây dựng tập train/test

- Dữ liệu sau khi tiền xử lý được lưu thành 1 file duy nhất. Mỗi dòng là 1 bài báo kèm theo thông tin danh mục của nó
- Đọc dữ liệu từ file và tách thành 2 list text (dữ liệu) và label (nhãn). Dữ liệu text[i] sẽ có nhãn là label[i]
- Chia làm 2 tập train (X_train, y_train) và test (X_test, y_test)
- Lưu train/test data ra file để sử dụng cho việc huấn luyện mô hình
- Đưa label về dạng vector để tiện cho tính toán sử dụng LabelEncoder

Load dữ liệu từ các file văn bản

```
def preProcess(sentences):


    text = [re.sub(r'([^\s\w]|_)+', '', sentence) for
      sentence in sentences if sentence!='']

    text = [sentence.lower().strip() for sentence in
      text]

    return text
```

```
def loadData(data_folder):
    texts = []
    labels = []
    #
    for folder in listdir(data_folder):
        #
        if folder != ".DS_Store":
            print("Load cat: ",folder)
            for file in listdir(data_folder + sep + folder):
                #
                if file!=".DS_Store":
                    print("Load file: ", file)
                    with open(data_folder + sep + folder + sep +  file, 'r', encoding="utf-8") as f:
                        all_of_it = f.read()
                        sentences  = all_of_it.split('.')

                        # Remove garbage
                        sentences = preProcess(sentences)
                        texts = texts + sentences
                        label = [folder for _ in sentences]
                        labels = labels + label
                        del all_of_it, sentences
    return texts, labels
```

Tokenizer các câu văn bản

Chuyển các câu văn bản từ dạng list of string về list of numbers.

['hôm nay chúng ta học xử lý ngôn ngữ tự nhiên']

=>

[20 1 22 31 34 22 1 12 67 43 22 14...]

```
def txtTokenizer(texts):
    tokenizer = Tokenizer(num_words=500)

    # fit the tokenizer on our text
    tokenizer.fit_on_texts([text.split() for text in texts])

    # get all words that the tokenizer knows
    word_index = tokenizer.word_index

    #return tokenizer, word_index
    tokenizer, word_index = txtTokenizer(texts)

    # put the tokens in a matrix
    X = tokenizer.texts_to_sequences(texts)
    X = pad_sequences(X)

    # prepare the labels (one hot vector)
    y = pd.get_dummies(labels)
```

Train model word2vec

- Sử dụng thư viện Gensim

```
word_model = gensim.models.Word2Vec(texts, size=300, min_count=1, iter=10)
word_model.save(data_folder + sep + "word_model.save")
```

- Ví dụ

```
print(word_model.wv.most_similar('cơm'))
```

[('nấu', 0.7218972444534302), ('cháo', 0.6976884603500366), ('nướng', 0.6886948347091675), ('bún', 0.679700255393981), ('phở', 0.645546197891235), ('riêu', 0.6107968091964722), ('nát', 0.6087987422943115), ('mì', 0.607367992401123), ('xào', 0.6070189476013184), ('rán', 0.5872985720634 46)]

Train model phân loại văn bản

```
model = Sequential()
model.add(Embedding(len(word_model.wv.vocab)+1,300,
        input_length=X.shape[1],weights=[embedding_matrix],trainable=False))
model.add(LSTM(300,return_sequences=False)) model.add(Dense(y.shape[1],activation="softmax"))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc’])
print(model.summary())
model.fit(X_train, y_train, epochs=350, batch_size=32, verbose=1, validation_data=(X_test, y_test))

model.evaluate(X_test,y_test)
```

Ex. Remove các loại dấu câu và tokenizer

```
def standardize_data(row):
    # remove stopword

    # Xóa dấu chấm, phẩy, hỏi ở cuối câu
    row = re.sub(r"[\.,\?]+$-", "", row)
    # Xóa tất cả dấu chấm, phẩy, chấm phẩy,
chấm than, ... trong câu
    row = row.replace(",", " ").replace(".", " ") \
        .replace(";", " ").replace("“", " ") \
        .replace(":", " ").replace("”", " ") \
        .replace("", " ").replace("", " ") \
        .replace("!", " ").replace("?", " ") \
        .replace("-", " ").replace("?", " ")
    row = row.strip()
    return row
```

```
from underthesea import word_tokenize
def tokenizer(row):
    return word_tokenize(row, format="text")
```

Ex. Embedding bằng tfidf

TF-IDF là một trong những kỹ thuật cơ bản trong xử lý ngôn ngữ giúp đánh giá mức độ quan trọng của một từ trong văn bản

- TF(Term frequency): Tần suất xuất hiện của 1 từ trong 1 document

  TF(t,d)= (Số lần xuất hiện từ t)/(Tổng số từ của d)

- IDF(Invert Document Frequency) : đánh giá mức độ quan trọng của 1 từ trong văn bản. Trong văn bản thường xuất hiện nhiều từ không quan trọng xuất hiện với tần suất cao
  - Từ nối: và, hoặc, …
  - Giới từ: ở, trong, của , để, ….
  - Từ chỉ định: ấy, đó, nhỉ, …

  IDF(t, D) = log_e(Số văn bản trong tập D /Số văn bản chứa từ t trong tập D)

- Những từ có tf-idf là những từ xuất hiện nhiều trong 1 văn bản này và xuất hiện ít trong văn bản khác

  TF-IDF(t, d, D) = TF(t, d) * IDF(t, D)

Ex. Embedding bằng tfidf

```
def embedding(X_train, X_test):
    global  emb
    emb = TfidfVectorizer(min_df=5,
     max_df=0.8,max_features=3000,sublinear_t
     f=True)
    emb.fit(X_train)
    X_train =  emb.transform(X_train)
    X_test = emb.transform(X_test)


    # Save pkl file
    joblib.dump(emb, 'tfidf.pkl')
    return X_train, X_test
```

```
emb = TfidfVectorizer(min_df=5,
    max_df=0.8,max_features=3000,
    sublinear_tf=True)
```

nghĩa là trong quá trình tạo Vocabulary chúng ta chỉ xét các từ có tần suất xuất hiện trên 5 lần trong tập các văn bản và dưới 80% trong toàn bộ các văn bản. Các từ trên 80% khả năng cao là ác stopwords

Ex. Train SVM

```
# 4. Embeding X_train
X_train,X_test  = embedding(X_train, X_test)


# 5. Train and save model
model = svm.SVC(kernel='linear', C = 1)
model.fit(X_train,y_train)
joblib.dump(model, 'saved_model.pkl')
```

Tài liệu tham khảo

- https://www.miai.vn/2020/05/04/nlp-series-1-thu-lam-he-thong-danh-gia-san-pham-lazada/
- https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/#ftoc-heading-11