**Các file:**

source:

* build\_features.py: đọc dữ liệu raw và chuyển thành dữ liệu dùng để train
* train\_model.py: train model từ dữ liệu đầu vào (sau khi build\_features)
* test\_model.py: để test (nhưng chưa viết gì)
* main.py: điền câu vào xem kết quả (dùng lúc chạy cuối cùng)
* NaiveBayes.py: class của NaiBay
* tokenize\_clean\_data: mấy hàm chuyển hóa câu thô thành token thành dữ liệu cho vào train được
* setup.py: chạy để cài đặt đủ thư viện sẽ sử dụng trong chương trình (nhưng chưa viết gì)
* utility.py: đây mới dùng để load dump ra file .pickle (có vẻ hơi thừa, về sau rút bớt đi)
* \_\_init\_\_.py: cái này để import các file .py khác trong cùng folder source này (rỗng nhưng không được xóa)

data:

* raw:
  + dataset.csv: bản full không che
  + demodataset.csv: thử ít ít data chạy thử xem mượt chưa
* processed:
  + before\_train.pickle: dữ liệu train sau khi build\_features chạy xong
  + test.pickle: dữ liệu test sau khi build\_features chạy xong
  + after\_train.pickle: model lưu lại sau khi train xong

notebook: (mấy cái ipynb để tham khảo sau xóa)

**Trình tự chạy chương trình: (cập nhật 12:30 12/11/2021)**

Bước 1: chạy build\_features.py

Bước 2: chạy train\_model.py

Bước 3: chạy main.py

**Trình tự build\_features.py:**

1. Đọc dữ liệu raw:

* Đọc dataset.csv, data có dạng [ label, time, date, query, username, text ]

Ví dụ 1 phần tử: ( "0", "1467810917", "Mon Apr 06 22:19:53 PDT 2009", "NO\_QUERY", "longtruong", "I work. He works. They work. We give up!" )

* Chuyển về dạng [ (text, label) ]

Ví dụ 1 phần tử: ( "I work. He works. They work. We give up!", 0 )

1. Cắt nhỏ kích thước dữ liệu ra (nhiều quá thì chạy lâu)
2. Token hóa và làm sạch dữ liệu

* Tokenize về dạng [ ([token], label) ]

Ví dụ 1 phần tử: ( [ "I", "work", ".", "He", "works", ".", "They", "work", ".", "We", "give", "up", "!" ], "0" )

* Làm sạch token [ ([cleaned\_token], label) ]

Ví dụ 1 phần tử: ( [ "work", "work", "work", "give", "up" ], 0 )

* Chuyển token về kiểu dict, key là từ (phân biệt các từ khác), value là số lần lặp lại trong câu đó 🡪 dạng [ ({cleaned\_token: time\_repeated}, label) ]

Ví dụ 1 phần tử: ( { "work": 3, "give": 1, "up": 1 }, 0 )

🡪 Dữ liệu này là dữ liệu sử dụng được trong lớp NaiBay (cả để train, test, classify)

1. Trộn, chia ra 2 tập để train và để test
2. Lưu lại các tập vào file before\_train.pickle và test.pickle tương ứng

**Trình tự train.py:**

1. Load file before\_train.pickle để lấy dữ liệu đầu vào train
2. Tạo đối tượng NaiBay và train

**Trình tự main.py:**

1. Load file after\_train.pickel để lấy kết quả đã được train
2. Token hóa và làm sạch câu (như trên) để đưa vào tính kết quả

**Giải nghĩa các thuộc tính class Naibay: (đặt tên hơi ngu nhưng có chú thích trong code, vẽ hình cho dễ hiểu này)**

* Giả sử có dữ liệu sau (làm tính toán thử thôi chứ méo có nghĩa đâu):

[ ( “walk work work hard”, 0 ),

( “work run bad”, 0 ),

( “worse walks wait”, 1 ),

( “wait good worked good good”, 1 ) ]

🡪 4 câu: 2 negative, 2 positive

* Làm sạch, ta được:

[ ( { 'walk': 1, 'work': 2, 'hard': 1 }, 0 ),

( { 'work': 1, 'run': 1, 'bad': 1 }, 0 ),

( { 'bad': 1, 'walk': 1, 'wait': 1 }, 1 ),

( { 'wait': 1, 'good': 3, 'work': 1 }, 1 ) ]

* Các thuộc tính của class NaiBay (trong hàm khởi tạo \_\_init\_\_(self)):

Có các từ phân biệt sau: walk, work, hard, run, bad, wait, good

🡪 distinctWords = [ “walk”, “work”, “hard”, “run”, “bad”, “wait”, “good” ]

🡪 countDistinctWords = 7 (là 7 từ phân biệt)

Tổng số lượng các từ (không phân biệt) là 15 từ (có tính cả số lần lặp)

🡪 countWords = 15

Danh sách label lần lượt là 0 , 0, 1, 1

🡪 labels = [ 0, 0, 1, 1 ]

🡪 distinctLabels = { 0: 2, 1: 2 } (0 xuất hiện 2 lần, 1 xuất hiện 2 lần)

Từ dữ liệu, ta có bảng sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| walk | work | hard | run | bad | wait | good |  |
| 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 3 | 1 |

🡪 phần màu trắng được lưu trong countWordInLine

Nhãn 0 có tổng cộng 7 từ, nhãn 1 có tổng cộng 8 từ (không phân biệt, tính cả lặp)

🡪 countAllWordByLabel = { 0: 7, 1: 8 }

Ví dụ từ work xuất hiện 3 lần trong nhãn 0, 1 lần trong nhãn 1. Tương tự với các từ khác, ta có (cái nào không xuất hiện trong 1 nhãn thì không ghi)

🡪 countWordByLabel = {

“walk”: { 0: 1, 1: 1 }

“work”: { 0: 3, 1: 1 }

“hard”: { 0: 1 }

“run”: { 0: 1 }

“bad”: { 0: 1, 1: 1 }

“wait”: { 1: 2 }

“good”: { 1: 3 }

}

Với nhãn 0, xác suất xuất hiện từ work tính theo công thức:

Table

Description automatically generated with low confidence

Tính tương tự với các từ và nhãn ta được

🡪 probWordByLabel = {

'walk': { 0: 0.142857, 1: 0.133333 },

'work': { 0: 0.285714, 1: 0.133333 },

'hard': { 0: 0.142857 },

'run': { 0: 0.142857 },

'bad': { 0: 0.142857, 1: 0.133333 }

'wait': { 1: 0.2 },

'good': { 1: 0.2666666 },

}

🡪 Tất cả phần này sẽ được tính toán trong hàm train(self, train\_data)

Phần classify tôi lười quá thêm sau nhé