1 Giới thiệu

Liệu rằng phần lời của một bài hát có đầy đủ thông tin để chúng ta phân loại chủ đề bài hát đó? Trong bài viết này, chúng ta sẽ dùng machine learning để trả lời câu hỏi này với các phương pháp Logistic Regression (PyTorch) / Naive Bayes / Genetic Algorithm / Decision Tree nhé!

2 Tải và xử lý dữ liệu

2.1 Trích xuất dữ liệu

Cảm ơn loibaihathot.com đã có một kho lời bài hát khá nhiều và dễ trích xuất. Tất cả những gì chúng ta cần chỉ là thư viện requests để lấy nội dung html của trang web. Và, BeautifulSoup để parse nội dung html ra element cho dễ trích xuất.

*# Import thư viện cần thiết*

**import** requests

**import** time

**import** csv

**import** re

**from** bs4 **import** BeautifulSoup

Xác định 1 số chủ đề có sẵn trên trang web để tài về. Base URL sẽ là url gốc, từ đây chúng ta replace {genre} để tải lời cho bài hát thuộc chủ đề tương ứng.

Lưu ý: Ở đây mình dùng từ genre không sát nghĩa “chủ đề” đâu nghen.

genre\_list = ['cach-mang', 'que-huong', 'thieu-nhi', 'tre']

base\_url = "https://loibaihathot.com/{genre}"

Tiếp theo, chúng ta tải hết tất cả các URL của các bài hát, phân loại theo chủ đề

lyric\_url\_list = {}

**for** genre **in** genre\_list:

url = base\_url.format(genre=genre)

html\_text = requests.get(url).text

soup = BeautifulSoup(html\_text, 'html.parser')

all\_links = list(map(**lambda** e: e['href'], soup.find\_all('a')))

lyric\_url\_list[genre] = list(set(filter(**lambda** url: '/20' **in** url, all\_links)))

Tạo thư mục data để lưu trữ dữ liệu, tránh việc phải chạy Trích xuất lại lần nữa:

**import** os

DATA\_FOLDER = 'data/'

**def** create\_path\_if\_nonexist(path):

**if** **not** os.path.exists(path):

os.mkdir(path)

create\_path\_if\_nonexist(DATA\_FOLDER)

Viết hàm để tải nội dung lời bài hát cho một bài hát, với input là đường dẫn đến bài hát đó:

**import** re

**def** download\_lyric(song\_url):

html\_text = requests.get(song\_url).text

soup = BeautifulSoup(html\_text, 'html.parser')

lyric = soup.find('div', class\_="entry-content content mt-6").get\_text(separator = '\n', strip = True)

lyric = re.sub('\n.+','',lyric, count=3).strip()

**return** lyric

Còn đây là hàm để tải lời nhiều bài cùng 1 lúc, hàm này sẽ dùng để phân luồng threading -> tăng tốc download:

**def** download\_songs(song\_urls, genre):

full\_path = os.path.join(DATA\_FOLDER, genre)

create\_path\_if\_nonexist(full\_path)

**for** url **in** song\_urls:

lyric = download\_lyric(url)

**if** len(lyric) < 10:

**continue**

file\_name =re.findall('/[^\/]+$', url)[0][1:-5]

**with** open(f'{full\_path}/{file\_name}.txt', 'w') **as** f:

f.write(lyric)

Với mỗi chủ đề, chúng ta sẽ dùng threading để tải hết lời bài hát về. Để tải nhanh hơn, chúng ta có thể chia thêm nhiều threads con nữa. Nhưng mình muốn mọi thứ đơn giản trước đã:

**import** threading

thread\_list = []

*# create list of threads*

**for** genre **in** genre\_list:

thread = threading.Thread(target=download\_songs, args=(lyric\_url\_list[genre],genre))

thread\_list.append(thread)

print("Download starting...")

*# start each thread*

**for** thread **in** thread\_list:

thread.start()

*# wait for all to finish*

**for** thread **in** thread\_list:

thread.join()

*# successfully excecuted*

print("Download finished!")

Download starting...

Download finished!

2.2 Tiền xử lý (Pre-processing)

Sau khi mình và train thử cho bộ dữ liệu ban đầu thì thấy độ chính xác đều dưới 20%. Kiểm tra lại thì thấy có nhiều bài download về vào sai thư mục chủ đề. Điều này có thể gây nhiễu dữ liệu, nên chúng ta sẽ xóa thủ xông những files sau để tăng độ chính xác khi huấn luyện:

*# Clean-up data*

to\_delete = {

"thieu-nhi":[

"cam-on-nguoi-da-roi-xa-toi.txt",

"chac-ai-do-se-ve.txt",

"dung-tin-em-manh-me.txt",

"hay-ra-khoi-nguoi-do-di.txt",

"khuon-mat-dang-thuong.txt",

],

"cach-mang":[

"cam-on-nguoi-da-roi-xa-toi.txt",

"chac-ai-do-se-ve.txt",

"dung-tin-em-manh-me.txt",

"hay-ra-khoi-nguoi-do-di.txt",

"khuon-mat-dang-thuong.txt",

],

"que-huong":[

"cam-on-nguoi-da-roi-xa-toi.txt",

"chac-ai-do-se-ve.txt",

"dung-tin-em-manh-me.txt",

"hay-ra-khoi-nguoi-do-di.txt",

"khuon-mat-dang-thuong.txt",

],

}

**for** subfolder, filenames **in** to\_delete.items():

**for** filename **in** filenames:

**try**:

filepath = f"{DATA\_FOLDER}{subfolder}/{filename}"

os.remove(filepath)

**except** Exception **as** e:

print(e)

Cài thêm thư viện underthesea để hỗ trợ tokenize ngôn ngữ Việt:

!pip install -q underthesea==1.3.5**a3**

Tiền xử lý lời bài hát, chủ yếu là xóa ký tự đặc biệt:

**def** preprocess\_lyric(text):

*# remove special characters*

**import** re

text = re.sub('[^\w\s]','', text).lower()

**return** text

preprocess\_lyric('Ngày mai??!! 13 em đi!')

'ngày mai 13 em đi'

3 Phương pháp 1: Logistic Regression với PyTorch

Dữ liệu text không thể huấn luyện trực tiếp được. Vì vậy chúng ta sẽ phải xây dựng features cho data. Ở đây, với mỗi lyrics, mình sẽ đếm tần suất của mỗi từ trong lyrics xuất hiện trong từng label (hay genre).

**import** numpy **as** np

n\_labels = len(genre\_list)

n\_features = 1 + n\_labels *# cộng 1 vì thêm bias*

Xây dựng từ điền freqs để đếm tần suất của mỗi từ theo từng label. freqs sẽ sử dụng để tạo features ở bước sau:

**import** glob

**from** underthesea **import** word\_tokenize

*# build freq dict with key pair as (word, label)*

freqs = {}

**for** label, genre **in** enumerate(genre\_list):

lyric\_folder = os.path.join(DATA\_FOLDER, genre)

**for** filepath **in** glob.glob(f"{lyric\_folder}/\*txt"):

**with** open(filepath, 'r') **as** file:

lyric = preprocess\_lyric(file.read())

lyric\_tokens = word\_tokenize(lyric)

**for** word **in** lyric\_tokens:

pair = (word, label)

**if** pair **not** **in** freqs:

freqs[pair] = 1

**else**:

freqs[pair] += 1

Viết hàm chuyển lời bài hát thành features.

**def** lyric\_to\_features(lyric\_raw, n\_labels):

lyric = preprocess\_lyric(lyric\_raw)

lyric\_tokens = word\_tokenize(lyric)

features = [0] \* n\_labels

**for** word **in** lyric\_tokens:

**for** label **in** range(n\_labels):

pair = (word, label)

count = freqs.get(pair, 0)

features[label] += count

features = [1] + features

*# print(features)*

**return** features

Đọc data từ ổ cứng và chuyển chúng thành features và labels:

data = [] *# features of songs*

**for** lyric\_label, genre **in** enumerate(genre\_list):

lyric\_folder = os.path.join(DATA\_FOLDER, genre)

**for** filepath **in** glob.glob(f"{lyric\_folder}/\*txt"):

**with** open(filepath, 'r') **as** file:

lyric\_raw = file.read()

features = lyric\_to\_features(lyric\_raw, n\_labels) + [lyric\_label]

data.append(features)

Sau đó, ta trộn và chia data thành X và y là những numpy array để dễ thao tác và train so với list truyền thống.

data = np.array(data)

np.random.shuffle(data)

y = data[:,-1]

X = data[:,:-1]

Trộn (shuffle) và chia data thành 2 tập train và test.

**import** torch

**import** matplotlib.pyplot **as** plt

**from** torch **import** nn

**from** torch **import** optim

**from** torchvision **import** datasets, transforms

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X\_train\_tensor = torch.from\_numpy(X\_train).float()

y\_train\_tensor = torch.as\_tensor(y\_train, dtype=torch.int64)

y\_train\_onehot\_tensor = torch.nn.functional.one\_hot(y\_train\_tensor, num\_classes=n\_labels).float()

3.1 Xây dựng mô hình

Chúng ta import những library cần thiết của PyTorch

**import** torch

**import** matplotlib.pyplot **as** plt

**from** torch **import** nn

**from** torch **import** optim

**from** torchvision **import** datasets, transforms

seed = 69

torch.manual\_seed(seed)

<torch.\_C.Generator at 0x7f4a9c335130>

Khai báo mô hình gồm 2 lớp: 1. Linear: Nhận input là tensor của bài hát được tokenize thành tensor có n\_features và output là tensor có số chiều là n\_labels tương ứng với số lượng chủ đề cần phân loại 2. LogSoftMax: Đây là dạng activation function cho bài toán phân loại đa lớp

model = nn.Sequential(nn.Linear(n\_features, n\_labels),nn.LogSoftmax(dim=1))

criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(model.parameters(), lr=0.001)

Hàm tính độ chính xác của mô hình:

**def** calculate\_accuracy(model, X, labels):

y\_hat = model(torch.as\_tensor(X).float())

preds = y\_hat.max(axis=1, keepdim=True)[1].numpy().squeeze()

correct = np.sum(preds == labels)

accuracy = correct / len(labels)

**return** accuracy

Tiến hành train model

epochs = 300

losses = []

accs = []

**for** e **in** range(epochs+1):

optimizer.zero\_grad()

*# tính y\_hat*

output = model(X\_train\_tensor)

*# tính loss*

loss = criterion(output, y\_train\_onehot\_tensor)

*# tính gradient*

loss.backward()

*# tối ưu gradient*

optimizer.step()

*# cập nhật loss*

running\_loss = loss.item()

losses.append(running\_loss)

*# tính accuracy*

acc = calculate\_accuracy(model, X\_test, y\_test)

accs.append(acc)

*# in thông số training*

**if** e % 100 == 0:

print(f"Training epoch {e} : loss: {running\_loss:.3f}; accuracy: {acc:.2%}")

Training epoch 0 : loss: 9201.797; accuracy: 40.00%

Training epoch 100 : loss: 16386.709; accuracy: 76.67%

Training epoch 200 : loss: 6421.723; accuracy: 56.67%

Training epoch 300 : loss: 906.633; accuracy: 86.67%

Mô hình chúng ta sau khi train 300 epoch có accuracy 86.67%.

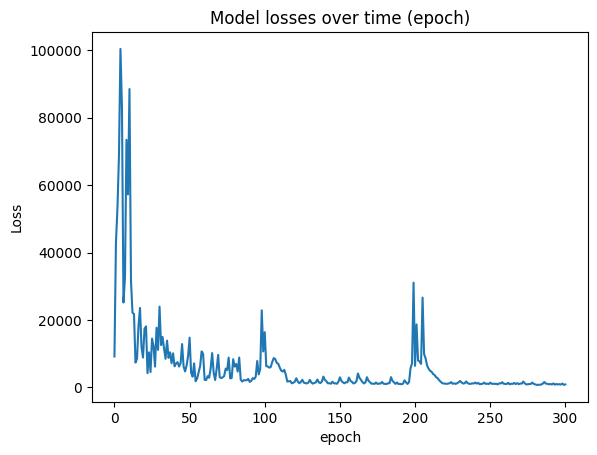
plt.plot(losses)

plt.xlabel('epoch')

plt.ylabel('Loss')

plt.title('Model losses over time (epoch)')

plt.show()



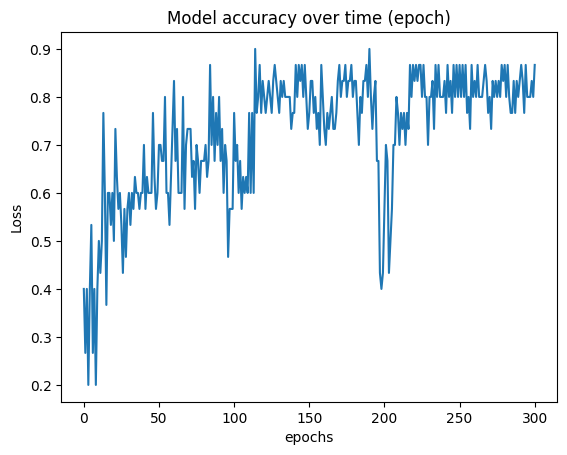
plt.plot(accs)

plt.xlabel('epochs')

plt.ylabel('Loss')

plt.title('Model accuracy over time (epoch)')

plt.show()



Hàm sau dùng để phân loại lời bài hát bất kỳ.

**def** predict\_label(model, lyric):

x = lyric\_to\_features(lyric, n\_labels)

y\_hat = model(torch.as\_tensor(np.array([x])).float())

pred = y\_hat.max(axis=1, keepdim=True)[1].numpy().squeeze()

**return** genre\_list[pred]

3.2 Predict

Ở đây chúng ta sẽ phân loại thử một số bài hát một ách cách thủ công nhé.

Đây là bài “Hát về anh”, thuộc thể loại cách mạng. Mô hình đã phân loại chính xác trường hợp này.

lyric = '''Hát Về Anh

Một ba lô, cây súng trên vai,

Người chiến sĩ quen với gian lao,

Ngày dài đêm thâu vẫn có những người lính trẻ,

Nặng tình quê hương canh giữ trên miền đất mẹ.

Rừng âm u, mây núi mênh mông

Ngày nắng cháy, đêm giá lạnh đầy.

Rừng mờ sương khuya bóng tối quân thù trước mặt,

Nặng tình non sông anh dâng trọn tuổi đời thanh xuân.

Cho em thơ ngủ ngon và vui bước sớm hôm đến trường

Cho yên vui mùa xuân đôi lứa còn hẹn hò ước mơ

Ðã có những hy sinh khó nói hết bằng lời

Nên đọng lại trong tôi những nghĩ suy.

Cho tôi bài ca về người chiến sĩ nơi tuyến đầu.

Nơi biên cường rừng sâu, anh âm thầm chịu đựng gió sương.

Đã có những gian lao, đã có những nhọc nhằn

Mang trong trái tim anh trọn niềm tin.

Xin hát mãi về anh người chiến sĩ biên cương

Xin hát mãi về anh người chiến sĩ biên cương

Nghe lời bài hát Hát Về Anh

Hát Về Anh '''

predict\_label(model, lyric)

'cach-mang'

Bài hát Dây Đủng Đỉnh Buồn được phân loại Quê Hương trong dữ liệu. Đây cũng là một dự đoán chính xác.

lyric = '''Dây Đủng Đỉnh Buồn (Remix)

Em đi theo chồng xa thôn làng cách biệt dòng sông.

Em đi theo chồng anh nơi này mỏi mon` đợi trông

Như dây đủng đỉnh nuôi trái tình bào tháng ngày qua.

Tình đã trọng xanh rồi người nỡ đem đi hái cho đành.

Ai xuôi chỉ mình ôm nỗi buồn cho người ta vui.

Ai xuôi chỉ mình xây duyên tình giờ đây lẻ loi.

Nhìn con nước chảy theo con thuyền lạc bến đời nhau.

Lời ước hẹn xưa giờ thì cũng xa xa cuối chân trời.

ĐK:

Đau thương thui thủi đêm trương, gió lạnh từng đêm lẻ bóng đơn côi.

Buồn miên man thầm trách cho đời lơ lửng chi rồi bỏ bạn mình ên.

Yêu thương xin trả cho người nuốt lệ nhìn theo đám cưới người ta.

Để bên đây đủng đỉnh u buồn, sao mang ân tình trao tặng người ta.

Nghe lời bài hát Dây Đủng Đỉnh Buồn (Remix)

Dây Đủng Đỉnh Buồn (Remix)'''

predict\_label(model, lyric)

'que-huong'

Bài hát thiếu nhi này cũng được phân loại đúng:

lyric = '''Ai yêu bác Hồ Chí Minh hơn thiếu nhi Việt Nam

Bác chúng em dáng cao cao, người thanh thanh

Bác chúng em mắt như sao, râu hơi dài

Bác chúng em nước da nâu vì sương gió

Bác chúng em thề cương quyết trả thù nhà

Hồ Chí Minh kính yêu, chúng em kính yêu Bác Hồ Chí Minh trọn một đời

Hồ Chí Minh kính yêu Bác đã bao phen bôn ba nước ngoài vì giống nòi

Bác nay tuy đã già rồi

Già rồi nhưng vẫn vui tươi

Ngày ngày chúng cháu ước mơ

Mong sao Bác sống muôn đời để dẫn dắt nhi đồng thành người và kiến thiết nước nhà bằng Người

Hồ Chí Minh kính yêu, chúng em kính yêu Bác Hồ Chí Minh trọn một đời

Hồ Chí Minh kính yêu, chúng em ước sao Bác Hồ Chí Minh sống muôn đời

Ai Yêu Bác Hồ Chí Minh Hơn Chúng Em Nhi Đồng'''

predict\_label(model, lyric)

'thieu-nhi'

Bài hát “Ánh nắng của anh” được phân loại chính xác cho thể loại “Trẻ”

lyric = '''Những phút giâу trôi qua tầm taу

Ϲhờ một ai đó đến bên anh

Lặng nghe những tâm tư nàу

Là tia nắng ấm

Là em đến bên anh cho vơi đi ưu phiền ngàу hôm qua

Nhẹ nhàng xóa đi bao mâу đen vâу quanh cuộc đời nơi anh

Phút giâу anh mong đến tình уêu ấу

Giờ đâу là em, người anh mơ ước bao đêm

Ѕẽ luôn thật gần bên em

Ѕẽ luôn là vòng taу ấm êm

Ѕẽ luôn là người уêu em

Ϲùng em đi đến chân trời

Lắng nghe từng nhịp tim anh

Lắng nghe từng lời anh muốn nói

Vì em luôn đẹp nhất khi em cười'''

predict\_label(model, lyric)

'tre'

Bây giờ mình sẽ lấy 2 bài hát không trong tập dữ liệu

Bài thứ nhất là một bài nhạc trẻ mới nổi gần đây “Anh Chưa Thương Em Đến Vậy Đâu”:

lyric = '''Sao mình không gạt bỏ đi hết những lời nói ngoài kia

Và sao mình không gạt bỏ đi hết những định kiến ngoài kia

Giữa ngân hà em biết đâu là

Biết đâu là thế gian này mà

Mình bên nhau, được yêu nhau, được trao nhau, tình yêu sâu trái tim đậm sâu

Giữa ngân hà em biết đâu là

Biết đâu một sớm mai khi mà

Cần bao lâu, chờ bao lâu, đợi bao lâu, tình trao nhau mãi thôi đậm sâu

Giữa ngân hà, giữa ngân hà, giữa ngân hà

Biết đâu là, biết đâu là, biết đâu là

Hành tinh của hai chúng ta

Một nơi của riêng chúng ta

Giữa ngân hà, giữa ngân hà, giữa ngân hà

Biết đâu là, biết đâu là, biết đâu là

Hành tinh của hai chúng ta

Ở 1 thế giới còn rất xa'''

predict\_label(model, lyric)

'tre'

Với bài hát “Bước qua mùa cô đơn”:

lyric = '''Mùa thu rơi vào em, vào trong giấc mơ hôm qua

Mùa thu ôm mình em, chạy xa vòng tay vội vã

Lời em nói ngày xưa đâu đây

Vẫn âm thầm chìm vào trong mây

Đến bao giờ, dặn lòng anh không mong nhớ

Mùa thu rơi vào em, vào trong chiếc hôn ngây thơ

Mùa thu không cần anh, vì em giờ đây còn mãi hững hờ

Ngày mai kia nếu có phút giây vô tình thấy nhau sẽ nói câu gì...

Hay ta chỉ nhìn

Lặng lẽ

Đi qua

Chào cơn mưa

Làm sao cứ kéo ta quay lại

Những rung động con tim

Lần đầu hai ta gặp gỡ'''

predict\_label(model, lyric)

'tre'

4 Phương pháp 2: Naive Bayes

Đây là phương pháp dựa vào xác suất thay vì đạo hàm như logistic regression.

Đầu tiên, chúng ta sẽ tạo dựng corpus của lời bài hát và label (thể loại) của chúng:

**import** glob

**from** underthesea **import** word\_tokenize

*# build freq dict with key pair as (word, label)*

corpus = []

labels = []

**for** label, genre **in** enumerate(genre\_list):

lyric\_folder = os.path.join(DATA\_FOLDER, genre)

**for** filepath **in** glob.glob(f"{lyric\_folder}/\*txt"):

**with** open(filepath, 'r') **as** file:

lyric = preprocess\_lyric(file.read())

corpus.append(lyric)

labels.append(label)

*# shuffle data*

**import** random

bundle = list(zip(corpus, labels))

random.shuffle(bundle)

corpus, labels = zip(\*bundle)

corpus = list(corpus)

labels = list(labels)

Với corpus và labels đã tìm built ở trên, chúng ta tiến hành xây dựng dictionary freqs. Cái này giúp tính xác suất likelihood ở những bước sau:

**def** build\_freqs\_dict(corpus, labels):

freqs = {}

**for** text, label **in** zip(corpus, labels):

**for** word **in** word\_tokenize(preprocess\_lyric(text)):

key\_pair = (word, label)

**if** key\_pair **not** **in** freqs:

freqs[key\_pair] = 1

**else**:

freqs[key\_pair] += 1

**return** freqs

Tiếp tục, xây dựng hàm để tính likelihood cho từng từ, phân loại theo labels:

**def** calc\_likelihood(freqs, labels):

vocab = set([key\_pair[0] **for** key\_pair **in** freqs.keys()])

V = len(vocab)

Ns = {}

unique\_labels = set(labels)

**for** label **in** unique\_labels:

Ns[label] = len([key\_pair **for** key\_pair **in** freqs **if** key\_pair[1]==label])

p\_likelihood = {}

**for** word **in** vocab:

**for** label **in** unique\_labels:

freq = freqs.get((word, label), 0)

N = Ns[label]

likelihood = (freq + 1) / (N + V)

**if** word **not** **in** p\_likelihood:

p\_likelihood[word] = {label: likelihood}

**else**:

p\_likelihood[word][label] = likelihood

**return** p\_likelihood

Tính xác suất Prior cho từng label:

**def** calc\_prior\_prop(y):

priors = {}

labels, counts = np.unique(y, return\_counts=True)

total = sum(counts)

**for** label, count **in** zip(labels, counts):

priors[label] = count / total

**return** priors

Và mô hình được “train” với hàm sau:

**def** train\_naive\_bayes(X\_train, y\_train):

freqs = build\_freqs\_dict(corpus=X\_train, labels=y\_train)

priors = calc\_prior\_prop(y\_train)

likelihood = calc\_likelihood(freqs, y\_train)

**return** priors, likelihood

Sau khi có đủ các hàm cần thiết, chúng ta bắt đầu split dữ liệu và train:

train\_size = int(len(labels) \* 0.8)

X\_train = corpus[:train\_size]

y\_train = labels[:train\_size]

X\_test = corpus[train\_size:]

y\_test = labels[train\_size:]

p\_priors, p\_likelihood = train\_naive\_bayes(X\_train, y\_train)

Tiếp tục, chúng ta viết hàm để phân loại cho một lyric cho trước. Lần đầu khi mình viết hàm này thì mình không dùng log, và kết quả là hàm lúc nào cũng trả về xác suất bằng 0. Sau một hồi debug thì mình phát hiện ra do likelihood của mỗi từ đều gần bằng 0, nên nếu nhân chúng lại với số lượng lớn thì kết quả lúc nào cũng bằng 0.

Thế nên, mình sử dụng hàm log để tránh tình trạng này:

**from** numpy **import** log

**def** predict\_naive\_bayes(lyric, p\_priors, p\_likelihood):

preds = {}

words = word\_tokenize(preprocess\_lyric(lyric))

*# print(words)*

**for** label, prior **in** p\_priors.items():

p = log(prior)

**for** word **in** words:

**if** word **in** p\_likelihood:

p += log(p\_likelihood[word][label])

preds[label] = p

predicted\_label = -1

predicted\_logprob = min(preds.values())

**for** label, prop **in** preds.items():

**if** prop > predicted\_logprob:

predicted\_label = label

predicted\_logprob = prop

**return** predicted\_label

Chúng ta thử tính xem mô hình có độ chính xác là bao nhiêu nhé:

*# test accuracy*

correct = 0

total = len(y\_test)

**for** text, label **in** zip(X\_test, y\_test):

pred = predict\_naive\_bayes(text, p\_priors, p\_likelihood)

correct += (pred == label)

print(f"Accuracy: {correct} / {total} = {correct/total:.2%}")

Accuracy: 16 / 30 = 53.33%

Sau vài lần chạy thì mô hình đạt độ chính xác 30 ~ 53.33%.

4.0.1 Một số ví dụ

lyric = '''Hát Về Anh

Một ba lô, cây súng trên vai,

Người chiến sĩ quen với gian lao,

Ngày dài đêm thâu vẫn có những người lính trẻ,

Nặng tình quê hương canh giữ trên miền đất mẹ.

Rừng âm u, mây núi mênh mông

Ngày nắng cháy, đêm giá lạnh đầy.

Rừng mờ sương khuya bóng tối quân thù trước mặt,

Nặng tình non sông anh dâng trọn tuổi đời thanh xuân.

Cho em thơ ngủ ngon và vui bước sớm hôm đến trường

Cho yên vui mùa xuân đôi lứa còn hẹn hò ước mơ

Ðã có những hy sinh khó nói hết bằng lời

Nên đọng lại trong tôi những nghĩ suy.

Cho tôi bài ca về người chiến sĩ nơi tuyến đầu.

Nơi biên cường rừng sâu, anh âm thầm chịu đựng gió sương.

Đã có những gian lao, đã có những nhọc nhằn

Mang trong trái tim anh trọn niềm tin.

Xin hát mãi về anh người chiến sĩ biên cương

Xin hát mãi về anh người chiến sĩ biên cương

Nghe lời bài hát Hát Về Anh

Hát Về Anh '''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'cach-mang'

Bài hát Dây Đủng Đỉnh Buồn được phân loại Quê Hương trong dữ liệu. Đây cũng là một dự đoán chính xác.

lyric = '''Dây Đủng Đỉnh Buồn (Remix)

Em đi theo chồng xa thôn làng cách biệt dòng sông.

Em đi theo chồng anh nơi này mỏi mon` đợi trông

Như dây đủng đỉnh nuôi trái tình bào tháng ngày qua.

Tình đã trọng xanh rồi người nỡ đem đi hái cho đành.

Ai xuôi chỉ mình ôm nỗi buồn cho người ta vui.

Ai xuôi chỉ mình xây duyên tình giờ đây lẻ loi.

Nhìn con nước chảy theo con thuyền lạc bến đời nhau.

Lời ước hẹn xưa giờ thì cũng xa xa cuối chân trời.

ĐK:

Đau thương thui thủi đêm trương, gió lạnh từng đêm lẻ bóng đơn côi.

Buồn miên man thầm trách cho đời lơ lửng chi rồi bỏ bạn mình ên.

Yêu thương xin trả cho người nuốt lệ nhìn theo đám cưới người ta.

Để bên đây đủng đỉnh u buồn, sao mang ân tình trao tặng người ta.

Nghe lời bài hát Dây Đủng Đỉnh Buồn (Remix)

Dây Đủng Đỉnh Buồn (Remix)'''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'que-huong'

Tuy nhiên, thì bài hát thiếu nhi này lại bị phân loại sai:

lyric = '''Ai yêu bác Hồ Chí Minh hơn thiếu nhi Việt Nam

Bác chúng em dáng cao cao, người thanh thanh

Bác chúng em mắt như sao, râu hơi dài

Bác chúng em nước da nâu vì sương gió

Bác chúng em thề cương quyết trả thù nhà

Hồ Chí Minh kính yêu, chúng em kính yêu Bác Hồ Chí Minh trọn một đời

Hồ Chí Minh kính yêu Bác đã bao phen bôn ba nước ngoài vì giống nòi

Bác nay tuy đã già rồi

Già rồi nhưng vẫn vui tươi

Ngày ngày chúng cháu ước mơ

Mong sao Bác sống muôn đời để dẫn dắt nhi đồng thành người và kiến thiết nước nhà bằng Người

Hồ Chí Minh kính yêu, chúng em kính yêu Bác Hồ Chí Minh trọn một đời

Hồ Chí Minh kính yêu, chúng em ước sao Bác Hồ Chí Minh sống muôn đời

Ai Yêu Bác Hồ Chí Minh Hơn Chúng Em Nhi Đồng'''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'cach-mang'

Bài hát “Ánh nắng của anh” được phân loại chính xác cho thể loại “Trẻ”

lyric = '''Những phút giâу trôi qua tầm taу

Ϲhờ một ai đó đến bên anh

Lặng nghe những tâm tư nàу

Là tia nắng ấm

Là em đến bên anh cho vơi đi ưu phiền ngàу hôm qua

Nhẹ nhàng xóa đi bao mâу đen vâу quanh cuộc đời nơi anh

Phút giâу anh mong đến tình уêu ấу

Giờ đâу là em, người anh mơ ước bao đêm

Ѕẽ luôn thật gần bên em

Ѕẽ luôn là vòng taу ấm êm

Ѕẽ luôn là người уêu em

Ϲùng em đi đến chân trời

Lắng nghe từng nhịp tim anh

Lắng nghe từng lời anh muốn nói

Vì em luôn đẹp nhất khi em cười'''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'tre'

Bây giờ mình sẽ lấy 2 bài hát không trong tập dữ liệu

Bài thứ nhất là một bài nhạc trẻ mới nổi gần đây “Anh Chưa Thương Em Đến Vậy Đâu”:

lyric = '''Sao mình không gạt bỏ đi hết những lời nói ngoài kia

Và sao mình không gạt bỏ đi hết những định kiến ngoài kia

Giữa ngân hà em biết đâu là

Biết đâu là thế gian này mà

Mình bên nhau, được yêu nhau, được trao nhau, tình yêu sâu trái tim đậm sâu

Giữa ngân hà em biết đâu là

Biết đâu một sớm mai khi mà

Cần bao lâu, chờ bao lâu, đợi bao lâu, tình trao nhau mãi thôi đậm sâu

Giữa ngân hà, giữa ngân hà, giữa ngân hà

Biết đâu là, biết đâu là, biết đâu là

Hành tinh của hai chúng ta

Một nơi của riêng chúng ta

Giữa ngân hà, giữa ngân hà, giữa ngân hà

Biết đâu là, biết đâu là, biết đâu là

Hành tinh của hai chúng ta

Ở 1 thế giới còn rất xa'''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'tre'

Với bài hát “Bước qua mùa cô đơn”:

lyric = '''Mùa thu rơi vào em, vào trong giấc mơ hôm qua

Mùa thu ôm mình em, chạy xa vòng tay vội vã

Lời em nói ngày xưa đâu đây

Vẫn âm thầm chìm vào trong mây

Đến bao giờ, dặn lòng anh không mong nhớ

Mùa thu rơi vào em, vào trong chiếc hôn ngây thơ

Mùa thu không cần anh, vì em giờ đây còn mãi hững hờ

Ngày mai kia nếu có phút giây vô tình thấy nhau sẽ nói câu gì...

Hay ta chỉ nhìn

Lặng lẽ

Đi qua

Chào cơn mưa

Làm sao cứ kéo ta quay lại

Những rung động con tim

Lần đầu hai ta gặp gỡ'''

genre\_list[predict\_naive\_bayes(lyric, p\_priors, p\_likelihood)]

'tre'

Khá thú vị rằng, mô hình Naive Bayes dù đạt độ chính xác thấp hơn Logistic Regression, nhưng khi test thủ công lại cho kết quả tương tự.

5 [Update] Phương pháp 3: Genetic Algorithm - Giải thuật di truyền

Đây cũng là một giải thuật thú vị cho bài toán phân loại. Ở đây mình xài thư viện pygad để tập trung vào việc chọn tham số thay vì code tay lại mô hình.

!pip install -q pygad

Viết hàm softmax dùng cho bài toán phân loại đa lớp:

**def** softmax(vec):

epsilon=1e-7

e\_vec = np.exp(np.clip(vec, a\_min=1e-7, a\_max=300))

**return** e\_vec / (np.sum(e\_vec) + epsilon)

Mình chưa tìm ra cách để pygad hỗ trợ matrix, nên dùng giải pháp truyền vào 1 vector, rồi sau đó khi tính predict thì sẽ reshape lại thành ma trận:

solution\_shape=(n\_features, n\_labels)

**def** predict(theta, X):

theta\_reshaped = np.reshape(theta, newshape=solution\_shape)

y\_hat = softmax(X @ theta\_reshaped).argmax(axis=1)

**return** y\_hat

Hàm fitness này chủ yếu là tính độ chính xác (accuracy):

**def** compute\_fitness(theta, theta\_idx):

predictions = predict(theta, X\_train)

corrects = np.sum(np.equal(predictions, y\_train))

**return** corrects / len(predictions)

Đây là tham số sẽ dùng để train. Mình phát hiện rằng sử dụng nhiều cá thể hơn và mating nhiều hơn thì mô hình sẽ hội tụ nhanh hơn:

fitness\_function = compute\_fitness

num\_generations = 100

sol\_per\_pop = 1500

num\_parents\_mating = int(.8 \* sol\_per\_pop)

num\_genes = n\_features \* n\_labels

init\_range\_low = -100

init\_range\_high = 100

*# parent\_selection\_type = "sss"*

parent\_selection\_type = "rank"

keep\_parents = int(.05 \* sol\_per\_pop)

keep\_elitism = int(.3 \* sol\_per\_pop)

crossover\_type = "single\_point"

mutation\_type = "random"

mutation\_percent\_genes = 80

Với pygad thì việc train rất đơn giản, chỉ là khởi tạo với tham số như trên rồi gọi hàm run():

**import** pygad

ga\_instance = pygad.GA(num\_generations=num\_generations,

save\_best\_solutions=True,

allow\_duplicate\_genes=False,

num\_parents\_mating=num\_parents\_mating,

fitness\_func=fitness\_function,

sol\_per\_pop=sol\_per\_pop,

num\_genes=num\_genes,

init\_range\_low=init\_range\_low,

init\_range\_high=init\_range\_high,

parent\_selection\_type=parent\_selection\_type,

keep\_parents=keep\_parents,

keep\_elitism=keep\_elitism,

crossover\_type=crossover\_type,

mutation\_type=mutation\_type,

mutation\_percent\_genes=mutation\_percent\_genes)

ga\_instance.run()

Hàm sau để tính accuracy trên tập test. Hàm fitness ở trên là ở tập train

**def** compute\_accuracy(theta):

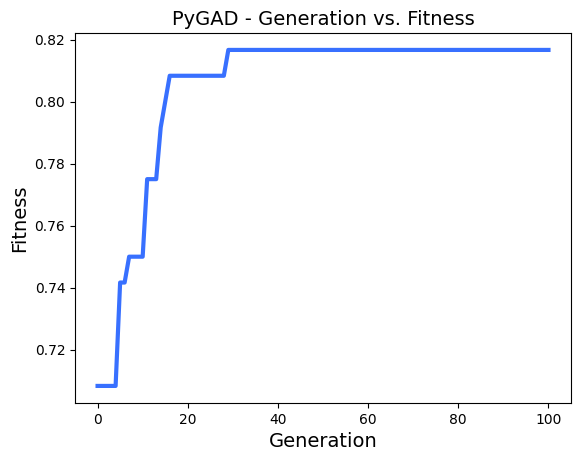
predictions = predict(theta, X\_test)

corrects = np.sum(np.equal(predictions, y\_test))

**return** corrects / len(predictions)

Mô hình hội tụ chỉ với hoảng 40 thế hệ:

fig = ga\_instance.plot\_fitness()



Vì accuracy trên tập train sẽ khác với tập test. Nên, chúng ta sẽ loop qua các nghiệm best rồi chọn ra cái có accuracy trên test cao nhất:

best\_solutions = ga\_instance.best\_solutions

accs = []

**for** sol **in** best\_solutions:

acc = compute\_accuracy(sol)

accs.append(acc)

best\_sol\_idx = np.argmax(accs)

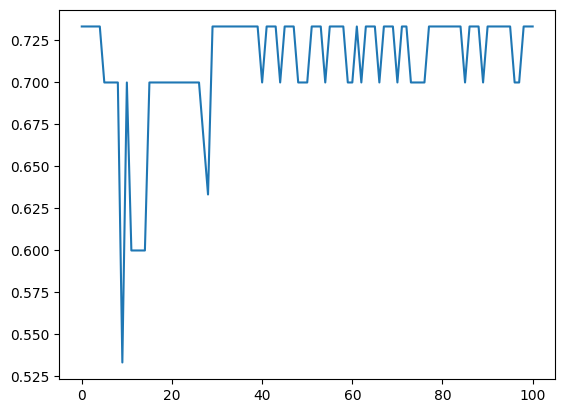
best\_sol = best\_solutions[best\_sol\_idx]

Đây là accuracy của các best solutions:

**import** matplotlib.pyplot **as** plt

plt.plot(accs)

plt.show()



Có thể thấy rằng sau gen thứ 40 thì acc nhảy qua lại giữa 0.7 và 0.725

Tiếp theo, hãy xem nghiệm best nhất có độ chính xác như thế nào nhé.

best\_sol\_acc = compute\_accuracy(best\_sol)

print(f"Accuracy of the best solution: {best\_sol\_acc:.2%}")

print(f"Best solution:\n", best\_sol.reshape(solution\_shape).round(4))

Accuracy of the best solution: 73.33%

Best solution:

[[ 45.0618 -71.966 6.4488 -33.7314]

[ 43.0644 61.1347 18.9067 -91.6843]

[-27.579 3.3492 -98.9866 73.9414]

[-24.5389 29.5234 -65.0851 -15.6497]

[ -4.3924 -23.3623 -70.5927 31.2352]]

6 [Update] Phương pháp 4: Decision Tree

Với SciKit-Learn thì việc train Decision Tree vô cùng đơn giản. Đầu tiên ta sẽ import mô hình và metrics

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn **import** metrics

Sau đó chỉ cần gọi hàm fit để train. Dùng predict để tính predictions trên tập test:

*# Create Decision Tree classifer object*

clf = DecisionTreeClassifier()

*# Train Decision Tree Classifer*

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

*#Predict the response for test dataset*

y\_pred = clf.predict(X\_test)

Chỉ với vài dòng code như vậy là bạn đã có một mô hình chính xác khá cao.

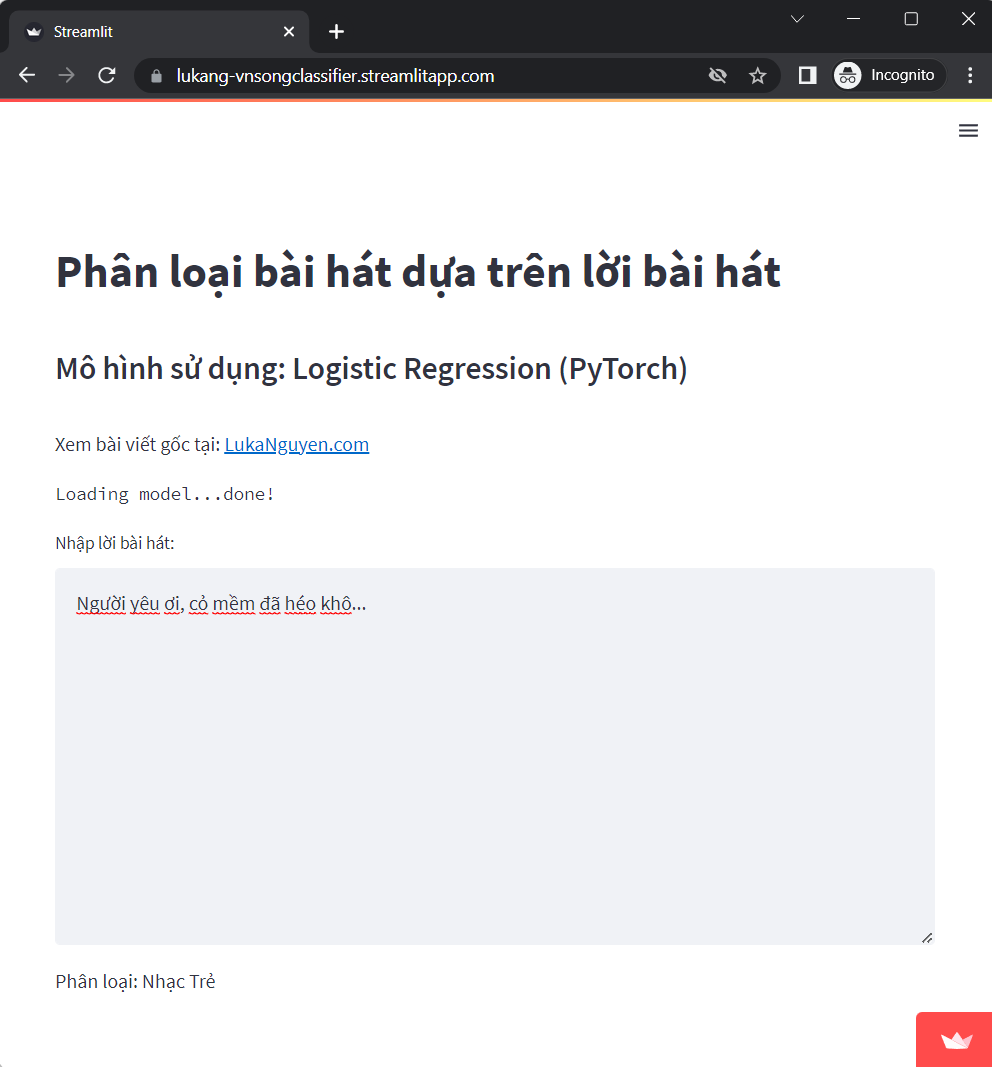
print(f"Decision Tree Accuracy: {metrics.accuracy\_score(y\_test, y\_pred):.2%}")

Decision Tree Accuracy: 70.00%

7 Kết luận

Trong project này, chúng ta đã cùng: 1. Trích xuất và xử lý dữ liệu lời bài hát 2. Xây dựng mô hình để phân loại bài hát dựa trên phần lời (lyrics) với hai mô hình phổ biến là Naive Bayes và Logistic Regression 3. So sánh độ chính xác của mô hình dựa trên tập test và cả test thủ công

Nếu bạn có hứng thú, hãy xem app demo ở đây nhé: [lukang-vnsongclassifier.streamlitapp.com](https://lukang-vnsongclassifier.streamlitapp.com/)



demo vietnamese song classifier