ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN TẤN BẢO – 17C 11002
TRẦN THÚY HIỀN – 17C 11026
NGUYỄN HÀ DUY PHƯƠNG – 17C 11 032

ĐỒ ÁN MÔN HỌC

XỬ LÝ NGÔN NGỮ NÓI

BÀI TẬP 4: ĐỌC CHỮ SỐ

GIÁO VIÊN

PGS. TS. Vũ Hải Quân

TP.HCM - 6/2018

MỤC LỤC

1 Mô tả bài toán

- Mục tiêu của bài tập là phát âm các kí số được nhập vào trong một tập tin.
- Đọc nội dung tập tin gồm các kí số, tìm kiếm trong bộ dữ liệu âm thanh cho trước, cắt ghép và tạo ra tập tin âm thanh tương ứng với nội dung đó.

2 Phương pháp làm

2.1 Tổng quát cách làm

Sử dụng tiếp tục kết quả bài 2 (nhận dạng rời rạc)

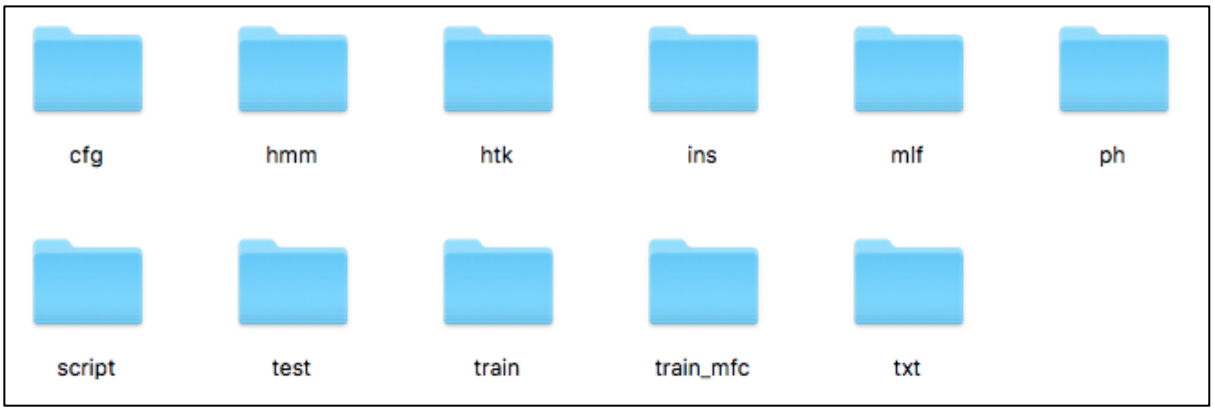
- Sử dụng HTK Toolkit của Đại học Cambridge và mô hình HMM.
- Tạo thư mục bài tập **htk** (Báo cáo giả định là thư mục chạy ngay trên ổ đĩa C)
- Cài đặt thêm môi trường **perl** và **python** cho một số bước phát sinh tập tên file...
- Sử dụng thêm từ điển **pydub** để đọc, cắt, ghép âm thanh. Để sử dụng pydub, cài đặt thêm **ffmpeg**.

Tổ chức dữ liệu gồm:

- *Train*: gồm tập file âm thanh **.wav** có trong thư mục *train/VIVOSPK14*.
- *Test*: gồm file tập tin **.txt** chứa các kí số trong thư mục *test*.

Cấu trúc thư mục **htk** có các mục sau:

- **cfg**: chứa các file config.
- **hmm**: chứa các thư mục hmm.
- **htk**: chứa các chương trình con của HTK.
- **ins**: chứa các file *.hed*, *.led*.
- **mlf**: chứa các file *.mlf*.
- **ph**: chứa các file *phones, fulllist*, *tiedlist*.
- **scripts**: chứa các file Perl (cần cài đặt môi trường cho Perl) và file Python.
- **test**: chứa tập tin kết qủa *number.txt* của bigram trên toàn bộ dữ liệu *vivos/test*, chứa thư mục test_mfc – gồm tất cả các tập tin test *.mfc*.
- **train**: chứa tập tin .wav làm dữ liệu huấn luyện.
- **train_mfc**: tất cả tập tin train *.mfc*.
- **txt**: tập tin từ điển, và một số tập tin cấu hình khác.



2.2 Các bước huấn luyện (mở cmd trỏ đến thư mục htk, lưu ý các file trong project này đều phải lưu dưới dạng ANSI):

Bước 1	Thực hiện theo các bước tương tự bài 1, để rút trích đặc trưng từ tập các file âm thanh **.wav**, tạo ra các **hmm**, và sử dụng hmm7 cho bước tiếp theo. Ở đây chỉ cần sử dụng tập tin **monophones1.txt**, vì chúng ta chỉ cần trích xuất được các khoảng thời gian của từng từ phát âm trong các tập tin âm thanh.

Bước 2	Gọi lệnh sau để tạo ra tập tin **aligned.out**.
	htk/HVite.exe -A -D -T 1 -l '*' -a -b silence -m -I mlf/words.mlf -H hmm/hmm7/macros -H hmm/hmm7/hmmdefs -i mlf/aligned.out txt/srcDict.txt ph/monophones1.txt -S txt/train_mfc.scp
	Tập tin aligned.out chỉ ra các segment của từng từ có trong các tập tin âm thanh, và có dạng như sau:
	`#!MLF!#`
	`"'*'/VIVOSSPK14_001.rec"`
	`0 4400000 sil -2217.223145 silence`
	`4400000 5200000 ng -785.442688 nguwowfi`
	`5200000 5500000 uwowf -286.038910`
	`5500000 6000000 i -394.035553`
	`6000000 6000000 sp -1.203973`
	`6000000 6800000 m -720.907837 mej`
	`6800000 7700000 ej -753.371460`
	`7700000 7700000 sp -1.203973`
	`7700000 8700000 gi -870.257935 giaf`
	`8700000 10300000 af -1118.573853`
	`10300000 10300000 sp -1.203973`
	Tập tin có cấu trúc theo dạng: tên file, danh sách các từ trong file, mỗi từ có kèm theo các thông số segment của từng âm vị trong từ đó.

Bước	Cài đặt mã nguồn để đọc tập tin **aligned.out**, sau đó tạo thành một mảng lưu trữ

3	***content_list*** tiện lợi cho việc dò tìm các từ trong bộ từ điển. Đoạn mã nguồn được xây dựng với ý tưởng như sau: - Đọc từng dòng tập tin **aligned.out** - Đọc và kiểm tra xem dòng đó có phải là tên file hay không? Nếu đúng, cắt và lưu lại đoạn tên chính xác. Ngược lại thực hiện tiếp. - Tách dòng văn bản ra thành mảng các từ theo dấu cách gọi là ***row***. Nếu là mảng có 5 kí tự chỉ ra đó là dòng bắt đầu của một từ mới. o Kiểm tra từ đó có trong mảng danh sách các từ chưa? Nếu chưa, thêm vào mảng, tạo mảng con của từ là danh sách các tập tin chứa từ đó. Ngược lại, thực hiện bước tiếp theo. o Khởi gán giá trị bắt đầu segment ***start*** của từ bằng phần tử đầu tiên trên trong mảng ***row***. o Kiểm tra tên file đang xét có trong mảng tập tin của từ đó chưa? Nếu chưa, thêm vào mảng. Ngược lại thực hiện bước tiếp theo. o Đọc tiếp các dòng, tìm cho đến dòng có chứa từ "*sp*", từ ngắt giữa các từ, thực hiện phân tích từ bằng dấu cách, lấy phần tử đầu tiên làm giá trị ***end*** – kết thúc segment. Lưu 2 giá trị ***start*** và ***end*** vào trong mảng tập tin trên. - Thực hiện đến khi tìm thấy dấu ".", kí tự chỉ kết thúc một tập tin âm thanh, quay lại bước đọc tên tập tin mới. - Ở đây, các từ sẽ giữ nguyên, có kí tự kết thúc "\n" do các từ nằm cuối dòng. Đoạn mã nguồn như sau:

```python
# Read file words.mlf
def readWords(file):
    filename = ""
    contentList = {}
    lines = file.readlines()
    for i in range(0, len(lines)):
        line = lines[i]
        if "'*'/" in line:
            filename = line[5:len(line)-5] + "wav"
        elif len(line) == 1 and '.' in line:
            continue
        elif '#!MLF!#' in line:
            continue
        else:
            row = line.split(' ')
            if len(row) == 5 and 'silence' not in line:
                start = row[0]
                end = row[0]
                letter = row[4]
                if letter not in contentList.keys():
                    contentList[row[4]] = {}
                if filename not in contentList[letter].keys():
                    contentList[letter][filename] = []
                for j in range(i+1, len(lines)):
                    spline = lines[j]
                    if 'sp' in spline:
                        end = spline.split(' ')[0]
                        i = j + 1
                        break

                contentList[letter][filename].append({'start': start, 'end': end})
            else:
                continue
    return contentList
```

Kết quả đạt được lưu vào tập tin **test/content.txt**. Việc lưu lại chỉ nhằm mục đích để tiện theo dõi cho việc thực hiện thử nghiệm.

```
mej
:{
VIVOSSPK14_001.wav: [{'start': '6000000', 'end': '7700000'}]
VIVOSSPK14_046.wav: [{'start': '21300000', 'end': '24400000'}]
VIVOSSPK14_099.wav: [{'start': '20800000', 'end': '24000000'}]
VIVOSSPK14_136.wav: [{'start': '5600000', 'end': '8700000'}]
VIVOSSPK14_188.wav: [{'start': '16100000', 'end': '18700000'}]
}
giaf
:{
VIVOSSPK14_001.wav: [{'start': '7700000', 'end': '10300000'}]
VIVOSSPK14_181.wav: [{'start': '15000000', 'end': '18700000'}]
}
```

Bước 4

Cài đặt tiếp mã nguồn cắt, ghép file âm thanh với các từ cần đọc:

- Duyệt qua từng từ cần đọc, tìm kiếm trong mảng ***content_list***.
- Nếu từ có trong bộ từ điển, lấy thông tin của từ đó trong mảng, ở đây chỉ cần: tên tập tin lưu từ đó đầu tiên, segment (***start***, ***end***) của từ trong tập tin đó.
- Ở đây, sử dụng thêm thư viện **pydub**, để đọc, cắt, ghép file. Thực hiện cắt đoạn âm thanh tương ứng với các giá trị lấy trên: new_audio =

new_audio[start:end], ***start*** và ***end*** ở đây được tính toán (chia cho 10000) lại để theo đơn vị milliseconds.

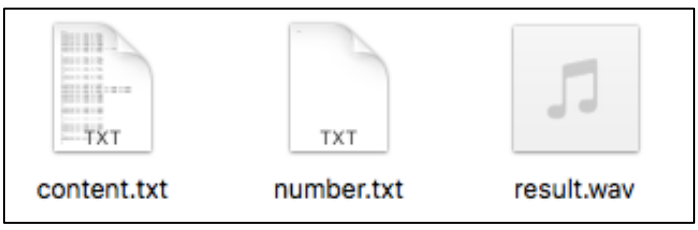
- Ghép lại bằng cách cộng ***new_audio*** vào biến ***combined_audio***, làm như vậy cho đến khi duyệt hết số từ cần đọc.
- combined_audio.export(result_audio_name, format="wav") gọi lệnh này để ghép xuất tập tin đã được ghép với tên mong muốn.

Đoạn mã nguồn như sau:

```
# create output wav file
def create_output_wav_file(contentList):
    i = 0
    newAudioList = []
    for letter in result_text_list:
        if letter in contentList.keys():
            wave_path = next(iter(contentList[letter]))
            time = list(next(iter(contentList[letter][wave_path])).values())
            start = int(int(time[0]) / 10000)
            end = int(int(time[1]) / 10000)
            newAudio = AudioSegment.from_wav(data_path + "/" + wave_path)
            newAudio = newAudio[start:end]
            if i == 0:
                combinedAudio = newAudio
            else:
                combinedAudio = combinedAudio + newAudio
            i += 1

    combinedAudio.export(result_audio_name, format="wav")
```

2.3 Các bước test:

Bước 1	- Chuẩn bị dữ liệu test là chuỗi gồm các kí số ví dụ "12345", lưu trong file **number.txt** - Tập tin **content.txt** để lưu mảng ***content_list*** như đã nói trên. - Tập tin **result.wav** là tập tin âm thanh đọc nội dung của **number.txt**. - Thư mục test như sau: 

Bước 2	Cài đặt đoạn mã nguồn đọc nội dung tập tin **number.txt**. Nếu là kí số thì trả về là từ tương ứng trong từ điển tiếng Việt telex. Ngược lại thì bỏ qua. Các từ trả về được nối thêm kí tự "/n" để tiện cho việc tìm kiếm trong ***content_list***. Lưu các từ vào trong mảng ***result_text_list*** cho thực hiện bước tạo đoạn âm thanh từng từ tương ứng.
Bước 3	Gọi lệnh này và truyền vào các tham số tương ứng: - Tập tin aligned.out - Tập tin chứa văn bản cần đọc - Thư mục chứa tập tin âm thanh - Tên tập tin kết quả - Kèm theo tên tập tin phụ ghi dữ liệu. python script/createContentWavFiles.py mlf/aligned.out test/number.txt train/VIVOSSPK14/ test/result.wav test/content.txt

3 Kịch bản thử nghiệm

Tiến hành kiểm thử với các nội dung văn bản như sau:

- 0123456789
- 1234
- 4567
- 814

Nội dung văn bản sau khi qua chuyển đổi từ kí số sang kí tự dạng như sau: khoong, moojt, hai, ba, boosn, nawm, sasu, bary, tasm, chisn.

Kết quả thử nghiệm được lưu tập tin **result.wav**

Bài tập này, chỉ kiểm tra kết quả đọc các kí số theo dạng từ kí số một.

4 Kết quả

4.1 Kết qủa thực nghiệm

Kết quả thử nghiệm đọc từng kí số có trong nội dung tập tin văn bản đầu vào cho ra chính xác tất cả các kí số cần đọc. Kết quả phát âm bỏ qua tất cả cá kí tự khác trong văn bản, chữ giữ lại và phát âm từng kí số.

5 Đánh giá cá nhân

- Quá trình huấn luyện và tạo ra tập tin chỉ các segment của từng từ trong tập tin âm thanh nhanh.
- Cài đặt thêm mã nguồn để đọc, cắt, ghép, tạo mảng bằng ngôn ngữ **Python**.

6 Tài liệu tham khảo

- Các bước thực hiện huấn luyện và kiểm thử phía trên được tham khảo chủ yếu (có chỉnh sửa, thêm hình ảnh và giải thích) từ file tham khảo ở htk_training_vn.pdf và htkbook.pdf
- https://github.com/jiaaro/pydub