TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC

TRẦN VĂN TRI - 0812543

NGUYỄN MINH TRÍ - 0812548

TRA TỪ ĐIỂN ANH VIỆT QUA CAMERA TRÊN ĐIỆN THOẠI DI ĐỘNG DÙNG ANDROID

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN

ThS. BÙI TẤN LỘC

PGS. TS. ĐINH ĐIỀN

KHÓA 2008- 2012

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

TpHCM, ngày ….. tháng …… năm ……

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

TpHCM, ngày ….. tháng …… năm ……

Giáo viên phản biện

LỜI CẢM ƠN

Chúng em xin gửi lời cám ơn sâu sắc đến thầy Đinh Điền và thầy Bùi Tấn Lộc là những người đã trực tiếp hướng dẫn chúng em, tạo nhiều điều kiện thuận lợi, góp ý kiến về mặt chuyên môn trong luận văn và nhờ đó mà chúng em mới có thể hoàn thành được luận văn trong thời gian cho phép.

Chúng con cũng xin gửi lời cám ơn đến cha mẹ và gia đình là những người thân nhất đã nuôi dưỡng, động viên, tạo điều kiện thuận lợi cho chúng con.

Chúng em xin cảm ơn các anh chị trong công ty Kim Từ Điển đã giúp đỡ, tạo điều kiện giúp chúng em hoàn thành luận văn này.

Đồng thời, chúng em cũng xin cám ơn chân thành đến quý thầy cô trong Khoa và các bạn bè gần xa đã luôn quan tâm và theo sát chúng em tạo cho chúng em nguồn động lực để hoàn thành luận văn.

Trong quá trình thực hiện luận văn có gì sai sót, kính mong nhận được sự chỉ bảo của quý thầy cô.

Tp Hồ Chí Minh, ngày … tháng … năm 2012

Nhóm sinh viên thực hiện

Trần Văn Tri – Nguyễn Minh Trí

Khoa Công Nghệ Thông Tin
Bộ môn Công Nghệ Tri Thức

ĐỀ CƯƠNG CHI TIẾT

Tên Đề Tài: Tra từ điển Anh Việt qua camera trên điện thoại dùng Android
Giáo viên hướng dẫn: PGS. TS. Đinh Điền – ThS. Bùi Tấn Lộc
Thời gian thực hiện: (từ ngày nhận đề tài đến ngày 25/6/2012)
Sinh viên thực hiện: Trần Văn Tri - 0812543 Nguyễn Minh Trí – 0812548
Loại đề tài: Xây dựng ứng dụng

Nội Dung Đề Tài: Xây dựng ứng dụng tra từ điển Anh-Việt trực tiếp trên điện thoại di động dùng hệ điều hành Android qua camera. Tìm hiểu bộ thư viện nhận dạng ký tự quang học Tesseract, cách thức chuyển mã Tesseract để chạy trên nền tảng Android. Tìm hiểu môi trường lập trình trên Android, các kỹ thuật xử lý ứng như thu nhận ảnh thông qua camera của điện thoại, sử dụng công cụ NDK để chạy mã nguồn C/C++. Tìm hiểu và cài đặt các thuật toán tra từ điển, cấu trúc lại tập tin dữ liệu từ điển, thuật toán khôi phục từ gốc Stemming và tìm từ gần đúng. Chương trình sau khi hoàn thiện sẽ bao gồm chức năng tra từ điển trực tiếp qua camera hoặc tra từ qua việc nhập liệu từ bàn phím.

Kế Hoạch Thực Hiện:

- 1/2/2012 – 29/2/2012:

  Trần Văn Tri: Tìm hiểu môi trường lập trình Android, tìm hiểu các thuật toán xây dựng cấu trúc dữ liệu từ điển.

  Nguyễn Minh Trí: Tìm hiểu môi trường lập trình Android, các thư

viện nhận dạng ký tự quang học OCR.

- 1/3/2012 – 31/3/2012:

  Trần Văn Tri: Cài đặt cấu trúc dữ liệu từ điển và tra từ trên Android.

  Nguyễn Minh Trí: Tìm hiểu thư viện Tesseract OCR, chuyển mã Tesseract và chạy thử nghiệm trên Android.

- 1/4/2012 – 29/4/2012:

  Trần Văn Tri: Tìm hiểu thuật toán khôi phục từ gốc và cài đặt trên Android.

  Nguyễn Minh Trí: Thiết kế kiến trúc cho ứng dụng, tích hợp các tính năng đã cài đặt bao gồm tra từ điển, bộ nhận diện Tesseract OCR, khôi phục từ gốc.

- 2/5/2012 – 31/5/2012:

  Trần Văn Tri: Thiết kế giao diện, các control, sửa lỗi trong chương trình.

  Nguyễn Minh Trí: Lập trình thu nhận ảnh và cải tiến chất lượng ảnh chụp trên điện thoại.

- 1/6/2012 – 24/6/2012

  Trần Văn Tri: Mã hóa dữ liệu từ điển, sửa lỗi chương trình. Viết báo cáo

  Nguyễn Minh Trí: Tích hợp thêm tính năng tìm từ gần đúng, sửa lỗi chương trình. Viết báo cáo.

Xác nhận của GVHD **Hướng dẫn chính** **Th.S Bùi Tấn Lộc** **Hướng dẫn phụ** **PGS.TS Đinh Điền**	**Ngày…….tháng……năm……** **SV thực hiện** **Trần Văn Tri – Nguyễn Minh Trí**

MỤC LỤC

LỜI CẢM ƠN ..... **iv**
ĐỀ CƯƠNG CHI TIẾT ..... **v**
MỤC LỤC ..... **viii**
DANH MỤC HÌNH ..... **xi**
DANH MỤC BẢNG ..... **xiii**
CÁC TỪ VIẾT TẮT ..... **xiv**
Chương 1 : TỔNG QUAN **1**
1.1. Bối cảnh và nhu cầu thực tế 1
1.2. Mục tiêu 3
1.3. Các đề tài liên quan 2
1.4. Nội dung khóa luận 4
Chương 2 : CÁC KỸ THUẬT CƠ BẢN TRÊN ANDROID **6**
2.1. Sơ lược về Android 6
2.1.1. Tổng quan 6
2.1.2. Các phiên bản Android 7
2.1.3. Kiến trúc và thiết kế 8
2.1.4. Máy ảo Dalvik 10
2.1.5. Android software development kit (SDK) 11
2.2. Native development kit (NDK) 12
2.2.1. Giới thiệu chung 12
2.2.2. Các hỗ trợ của NDK 13
2.2.3. Sử dụng NDK 13
2.2.4. Nội dung của bộ NDK 14
2.2.5. Giới thiệu về JNI – Java native interface 15
Chương 3 : NHẬN DẠNG KÝ TỰ QUANG HỌC **18**
3.1. Giới thiệu chung 18
3.1.1. Sơ lược về nhận dạng ký tự quang học – OCR 18
3.1.2. Các phương pháp áp dụng OCR trong luận văn 18

3.1.3. So sánh các thư viện / công cụ nhận dạng ký tự quang học 20
3.1.4. Kết luận 21
3.2. Giới thiệu về bộ nhận dạng ký tự quang học Tesseract 22
3.2.1. Lịch sử 22
3.2.2. Kiến trúc hoạt động 24
3.2.3. Cài đặt và sử dụng thư viện Tesseract trên Android 25
3.2.4. Huấn luyện dữ liệu trên Tesseract 29
3.2.5. Quá trình huấn luyện ngôn ngữ và font mới 30
Chương 4 : TRA TỪ ĐIỂN ANH-VIỆT 35
4.1. Tổng quan 35
4.2. Khôi phục từ gốc (Stemming) 37
4.3. Tìm từ gần đúng 40
4.3.1. Khoảng cách Levenstein 40
4.3.2. Thay thế các ký tự gần đúng 42
4.4. Cấu trúc dữ liệu từ điển 43
4.4.1. Tổ chức các mục từ có cùng kích thước cố định. 44
4.4.2. Tổ chức các mục từ có kích thước biến động. 44
4.4.3. Tổ chức dữ liệu từ điển tra cứu nhanh 45
Chương 5 : CÀI ĐẶT VÀ THỰC NGHIỆM ỨNG DỤNG 50
5.1. Vẽ khung và các control trên màn hình camera. 50
5.2. Thu nhận ảnh từ camera điện thoại. 52
5.3. Hiển thị tiếng Việt và định dạng chữ trên màn hình. 55
5.3.1. Hiển thị tiếng Việt trên Android 56
5.3.2. Định dạng ngữ nghĩa từ điển. 57
5.4. Mã hóa dữ liệu từ điển. 61
5.5. Lưu trữ cấu hình chức năng của ứng dụng. 63
5.6. Kỹ thuật phát âm từ tiếng Anh dùng API trên Android 66
5.7. Môi trường phát triển ứng dụng 68
5.8. Hướng dẫn cài đặt và sử dụng 69
5.8.1. Cài đặt chương trình 69
5.8.2. Hướng dẫn sử dụng 70
5.9. Kết quả thử nghiệm 74

5.9.1. Thử nghiệm khối nhận dạng ký tự 74
5.9.2. Thử nghiệm khối xử lý ngôn ngữ 76
5.9.3. Đánh giá kết quả 78
5.9.4. So sánh ứng dụng với các ứng dụng hiện có trên thị trường 78
TỔNG KẾT 81
- **MỘT SỐ KẾT QUẢ ĐẠT ĐƯỢC** 81
- **HẠN CHẾ** 82
- **HƯỚNG PHÁT TRIỂN** 82

TÀI LIỆU THAM KHẢO 84

DANH MỤC HÌNH

Hình 1.1 Sơ đồ khối tổng quát của chương trình 4
Hình 2.1 Điện thoại dùng hệ điều hành Android 6
Hình 2.2 Kiến trúc tổng thể của Android [1] 8
Hình 2.3 Cơ chế hoạt động của máy ảo Dalvik và Java 11
Hình 2.4 Minh họa trình giả lập điện thoại Android 12
Hình 2.5 JNI đóng vai trò trung gian trong việc giao tiếp giữa C/C++ và Java 15
Hình 2.6 Nội dung tập tin cấu hình biên dịch trong JNI 16
Hình 3.1 Quá trình thực hiện OCR 18
Hình 3.2 Sơ đồ khối nhận diện ký tự quang học trong chương trình 20
Hình 3.3 Kiến trúc tổng thể của Tesseract [2] 25
Hình 3.4 Minh họa cấu trúc của project tesseract-android-tools 26
Hình 3.5 Minh họa một phần các chỉ thị để biên dịch mã nguồn thư viện C/C++ trong tập tin Android.mk 27
Hình 3.6 Quá trình sử dụng NDK để biên dịch thư viện C/C++ trên Android 28
Hình 3.7 Quá trình biên dịch mã nguồn thư viện Tesseract thành công trên Android 28
Hình 3.8 Cấu trúc tập tin dạng hộp 32
Hình 3.9 Quá trình huấn luyện dữ liệu trên Tesseract 34
Hình 4.1 Sơ đồ thuật toán tra từ điển và xử lý ngôn ngữ tự nhiên 36
Hình 4.2 Sơ đồ thuật toán khôi phục từ gốc 39
Hình 4.3 Sơ đồ tổ chức tập tin từ điển 48
Hình 5.1 Giao diện màn hình camera 50
Hình 5.2 Minh họa gia đình font Droid 56
Hình 5.3 Hình Định dạng văn bản hiển thị theo các kiểu phong cách 58
Hình 5.4 Hình định dạng liên kết 60
Hình 5.5 ScreenPreference 65
Hình 5.6 ListPreference 65
Hình 5.7 Biểu tượng chương trình sau khi cài đặt hoàn tất 70

Hình 5.8 Màn hình chương trình khi khởi động 70
Hình 5.9 Màn hình hiển thị nghĩa của từ 71
Hình 5.10 Màn hình với hệ thống menu setting ở bên dưới 72
Hình 5.11 Màn hình tra từ điển theo cách thông thường 73
Hình 5.12 Màn hình thiết lập setting 73
Hình 5.13 Kết quả trước khi tra từ 76
Hình 5.14 Màn hình hiển thị nghĩa của từ sau khi xử lý từ gốc 77
Hình 5.15 Màn hình hiển thị danh sách từ gần đúng với kết quả nhận dạng 77

DANH MỤC BẢNG

Bảng 3.1 So sánh phần mềm thương mại và Tesseract 23
Bảng 3.2 Độ chính xác của Tesseract trên một số ngôn ngữ 23
Bảng 4.1 Minh họa ma trận kết quả sau khi tính khoảng cách Levenstein 41
Bảng 4.2 Bảng mô tả các trường dữ liệu 43
Bảng 5.1 Kết quả thử nghiệm bộ nhận dạng trong chương trình 74
Bảng 5.2 Một số kết quả nhận diện sai 75
Bảng 5.3 Đánh giá tốc độ thực thi của chương trình 78
Bảng 5.4 Bảng so sánh ứng dụng với Camera Dictionary 79
Bảng 5.5 Các tính năng chính trong chương trình 81

CÁC TỪ VIẾT TẮT

OCR	Optical Character Recognition
JNI	Java Native Interface
SDK	Software Development Kit
NDK	Native Development Kit
API	Application Programming Interface
DES	Data Encryption Standard
TTS	Text To Speech
CMD	Command Line
UNLV	University of Nevada-Las Vegas

Chương 1 : TỔNG QUAN

1.1. Bối cảnh và nhu cầu thực tế

Trong thời buổi công nghệ thông tin phát triển như vũ bão, các thiết bị điện tử ngày càng phát triển vượt bậc điển hình là các dòng máy tính, laptop, điện thoại di động đã trở nên phổ biến, ngày càng mạnh mẽ và nhỏ gọn phục vụ cho nhu cầu trao đổi thông tin liên lạc giữa mọi người. Trong đó điện thoại là một vật không thể thiếu trong đời sống con người và ngày càng có sự phát triển vượt bậc. Từ đó dẫn đến việc hình thành các dòng điện thoại thông minh - smartphone được tích hợp nhiều chức năng và kích thước càng ngày càng nhỏ gọn. Đáp ứng xu thế phát triển đó, các dòng điện thoại thông minh đã ra đời với cấu hình mạnh mẽ và nhiều tính năng hữu ích đang dần chiếm hữu thị trường.

Bên cạnh đó, nhu cầu về từ điển để phục vụ cho mọi người trong việc học tập, giao tiếp… cũng trở nên cần thiết. Chính vì thế nhiều chương trình từ điển ngôn ngữ đã được ra đời trên các nền tảng của thiết bị di động để phục vụ cho nhu cầu đó. Tuy nhiên các chương trình từ điển phần lớn yêu cầu người sử dụng phải nhập từ trực tiếp trên bàn phím điện thoại sau đó mới thực hiện việc tra từ. Đối với các ngôn ngữ ký tự Latinh thì việc nhập và tra từ sẽ dễ dàng hơn nhưng đối với các ngôn ngữ khác như tiếng Trung hoặc tiếng Nga chẳng hạn thì việc sử dụng từ điển bằng cách nhập từ vào và tra sẽ khó khăn hơn cho cho người sử dụng đòi hỏi người dùng phải biết rõ mẫu tự của ngôn ngữ đó nhưng đối với những người chưa biết hoặc chỉ mới làm quen với các ngôn ngữ này thì việc nhập từ sẽ rất khó khăn. Thí dụ như trong trường hợp một người đi du lịch qua đất nước khác nhưng không biết hoặc biết rất ít về ngôn ngữ đó thì sẽ khó khăn khi nhập từ để tra nghĩa. Vậy nên nếu phát triển một ứng từ điển nhưng không bắt buộc người dùng phải nhập từ vào mà cho phép người dùng có thể tra từ một cách gián tiếp thông qua camera của thiết bị điện thoại thì rõ ràng sẽ tiện lợi hơn rất nhiều. Vì phần lớn các dòng điện thoại thông minh hiện nay đều được trang bị camera nên việc phát triển một ứng dụng tra từ qua camera sẽ trở nên cần thiết hơn và phù hợp với tình hình thực tế.

Hiện nay các dòng điện thoại thông minh chạy trên nhiều nền tảng khác nhau. Trong đó nổi lên hai nền tảng chính đang chiếm lĩnh thị trường di động hiện nay là iOS của Apple và Android của Google. Hệ điều hành di động Android của Google đang cạnh tranh với iOS và có số lượng thiết bị lớn hơn với nhiều hãng sản xuất và mẫu mã đa dạng.

Gắn nhu cầu thực tế trong việc tra từ điển sử dụng camera trên điện thoại cùng với nền tảng Android đang được sử dụng phổ biến hiện nay nên nhóm chúng em quyết tâm xây dựng chương trình tra từ điển Anh Việt trực tiếp qua camera trên điện thoại Android.

1.2. Các đề tài liên quan

Luận văn tập trung vào phát triển ứng dụng phục vụ tra từ điển Anh Việt qua camera trên điện thoại Android. Đề tài này cũng dựa trên hướng nghiên cứu về tra từ điển qua camera trên điện thoại [2] – Luận văn thạc sĩ của anh Nguyễn Hoàng Giang. Ngoài ra trên thị trường cũng xuất hiện nhiều phần mềm có chức năng nhận diện từ qua camera điện thoại. Tiêu biểu cho môi trường Android là ứng dụng CamDictionary.

- Đề tài "Xây dựng ứng dụng tra từ điển bằng camera trên điện thoại di động" **Error! Reference source not found.** là đề tài có nhiều nét tương đồng với đề tài mà chúng em đang thực hiện. Cả hai đề tập trung giải quyết vấn đề là cách thức tra từ mới bằng cách dùng camera để nhận diện từ và sau đó tra từ điển. Cả hai đề tài đều sử dụng bộ Tesseract làm thành phần nhận diện ký tự quang học chính của chương trình. Chi tiết về Tesseract sẽ được giới thiệu chong chương 3 của báo cáo. Điểm khác biệt duy nhất của 2 đề tài là nền tảng phát triển. Đề tài đầu tiên sử dụng tra từ điển trên hệ điều hành Symbian của Nokia còn đề tài trong luận văn sử dụng Android của Google làm nền tảng chính.
- Ứng dụng CamDictionary: Đây là một ứng dụng trên Android dùng để dịch từ qua camera điện thoại, có thể xem là chương trình gần tương đương với

luận văn nhất. Ứng dụng do công ty Insig[1] phát triển. Đây là một công ty của Mỹ chuyên cung cấp các ứng dụng thực hiện việc nhận diện ký tự quang học trên nhiều môi trường như: máy quét, điện thoại, máy tính bảng… Ứng dụng CamDictionary sử dụng camera của điện thoại để nhận diện ký tự hoặc một đoạn câu sau đó dùng tính năng Google Translate qua môi trường mạng để dịch từ hoặc đoạn câu đó. Bộ nhận diện ngôn ngữ của chương trình khá chính xác nhưng việc tra cứu và dịch từ phụ thuộc vào kết nối mạng và độ chính xác của bộ dữ liệu từ điển Google. Chi tiết kết quả so sánh tính năng của luận án và chương trình CamDictionary sẽ được giới thiệu trong phần tổng kết của báo cáo.

1.3. Mục tiêu

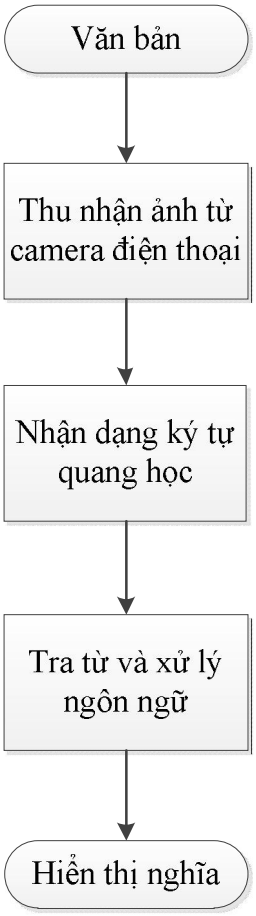
Mục tiêu của đề tài là xây dựng một ứng dụng trên điện thoại di động sử dụng camera để quét hình ảnh và sử dụng bộ nhận diện ký tự quang học (Optional Character Recognition – OCR) để rút trích ra các từ trong hình ảnh. Từ đó làm dữ liệu đầu vào cho việc tra từ.

Để xây dựng được ứng dụng tra từ điển qua camera, luận văn sẽ tập trung giải quyết các vấn đề sau:

- Tìm hiểu về môi trường lập trình trên nền tảng Android.
- Tìm hiểu sâu việc lập trình thu nhận ảnh từ camera của điện thoại.
- Tìm hiểu về bài toán nhận dạng ký tự quang học và cách sử dụng thư viện Tesseract OCR đồng thời tìm hiểu cách thức biên dịch mã nguồn thư viện Tesseract để chạy trên môi trường Android.
- Nghiên cứu xây dựng cấu trúc dữ liệu để thực hiện việc tra từ.
- Tìm hiểu các thuật toán xử lý ngôn ngữ để tăng khả năng tra từ chính xác cho ứng dụng như khôi phục từ gốc, tra từ gần đúng và áp dụng các thuật toán đó vào trong chương trình.
- Xây dựng ứng dụng hoàn chỉnh với đầy đủ các chức năng đã đề ra đồng thời cải tiến thêm các tính năng mới trong chương trình.

[1] http://www.intsig.com/en/index.html

Chương trình được chia làm 3 phần chính đó là thu nhận ảnh của văn bản từ camera điện thoại, nhận dạng ký tự quang học, phần tra từ và xử lý ngôn ngữ.



Hình 1.1 Sơ đồ khối tổng quát của chương trình

1.4. Nội dung khóa luận

Nội dung của luận văn bao gồm 6 chương:

Chương 1. Mở đầu: Bối cảnh và nhu cầu thực hiện đề tài, mục tiêu của đề tài và nội dung của khóa luận.

Chương 2. Kỹ thuật lập trình cơ bản trên Android: sơ lược về Android, lập trình truy xuất camera, sử dụng công cụ NDK biên dịch mã nguồn trên Android.

Chương 3. Nhận dạng ký tự quang học OCR: Giới thiệu chung về nhận dạng ký tự quang học, bộ nhận dạng ký tự quang học Tesseract và cách huấn luyện dữ liệu.

Chương 4. Tra từ điển Anh Việt: Cấu trúc dữ liệu từ điển, khôi phục từ gốc và tra từ gần đúng.

Chương 5. Cài đặt thực nghiệm: kết quả thử nghiệm và đánh giá chương trình.

Kết luận: Hạn chế của luận văn và hướng phát triển trong tương lai.

Chương 2 : CÁC KỸ THUẬT CƠ BẢN TRÊN ANDROID

2.1. Sơ lược về Android

2.1.1. Tổng quan

Android là hệ điều hành mở dựa trên nền tảng Linux dùng cho các thiết bị di động bao gồm điện thoại thông minh, máy tính bảng, máy tính xách tay. Được phát triển ban đầu tại công ty liên hợp Android sau đó công ty này được Google mua lại vào năm 2005 và biến Android thành một hệ điều hành mở trên các thiết bị di động.

Android chính thức ra mắt vào ngày 5/11/2007 cùng với sự ra đời của liên minh thiết bị cầm tay mở OHA (Open Handset Alliance). Liên minh OHA là một tổ chức bao gồm khoảng hơn 78 công ty viễn thông, di động và phần cứng như Goolge, Sony Ericsson, Samsung, Nvidia, Qualcomm… Mục tiêu của hội này là phát triển các chuẩn mở chung cho thiết bị di động trong tương lai. Và Android là sản phẩm chủ lực của hãng. Mã nguồn của Android là mã nguồn mở và được công bố dưới dạng giấy phép Apache.

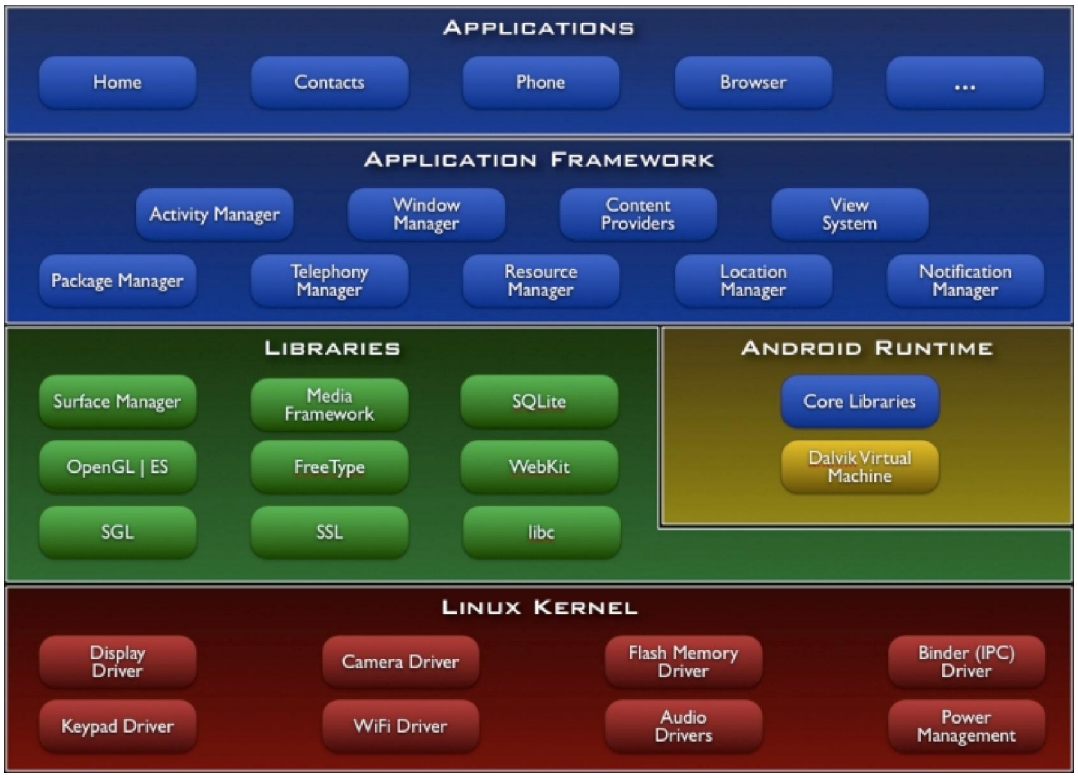
Hình 2.1 Điện thoại dùng hệ điều hành Android

2.1.2. Các phiên bản Android

Từ lúc ra đời đến nay, Android đã tung ra nhiều phiên bản khác nhau với những nâng cấp và cải tiến theo từng phiên bản. Sau đây là danh sách các phiên bản Android hiện có:

- Phiên bản 1.5 (Cupkake): phiên bản chính thức dầu tiên của Android trên điện thoại.
- Phiên bản 1.6 (Donut).
- Phiên bản 2.0/2.1 (Eclaire).
- Phiên bản 2.2 (Froyo).
- Phiên bản 2.3 (Gingerbread).
- Phiên bản 3.0 / 3.1 (Honeycomb): phiên bản dành riêng cho máy tính bảng (tablet).
- Phiên bản 4.0 / 4.0.1 / 4.0.3 (Icecream sandwich): đây là phiên bản Android mới nhất hiện nay và được dùng trên các điện thoại thông minh và cả máy tính bảng.

2.1.3. Kiến trúc và thiết kế



Hình 2.2 Kiến trúc tổng thể của Android [1]

Nhìn vào kiến trúc của Android thì hệ điều hành Android được chia thành các tầng như trong hình bao gồm: Applications, Application Framework, Libraries, Android Runtime, Linux Kernel. Trong đó 2 tầng Applications và Application Framework được viết bằng ngôn ngữ Java. Còn các tầng từ Libraries đến Linux Kernel được viết bằng ngôn ngữ C/C++ hay còn gọi là mã gốc - native code.

- Tầng **Applications**: Đây là tầng cao nhất trong hệ điều hành Android. Tầng này bao gồm các ứng dụng được viết và cài đặt sẵn như: lịch, trình duyệt web, danh bạ, camera… Các ứng dụng tại tầng này đều được viết bằng ngôn ngữ Java.
- Tầng **Application Framework**: Bên dưới tất cả các ứng dụng là một tập hợp các dịch vụ và hệ thống cho phép các nhà phát triển phần mềm có thể gọi các hàm hỗ trợ sẵn qua giao diện lập trình ứng dụng API (Application Programming Interface).

- Tập hợp các màn hình (Views) mở rộng dùng để xây dựng nên giao diện chương trình như nút bấm, danh sách, hộp thoại, text box, các sự kiện…
- Bộ cung cấp nội dung (Content Provider): Cung cấp khả năng truy xuất và chia sẽ dữ liệu giữa các ứng dụng.
- Quản lý tài nguyên (Resouce Manager): Quản lý các loại tập tin không phải là mã nguồn. Cung cấp khả năng truy cập đến các tài nguyên khác trong ứng dụng như các chuỗi, tập tin đồ họa, các tập tin định dạng giao diện (layout).
- Quản lý thông báo (Notification Manager): Quản lý và hiển thị các thông báo ở thanh trạng thái (Status Bar).
- Quản lý hoạt động (Activity manager): Quản lý vòng đời và chu trình hoạt động của các ứng dụng.

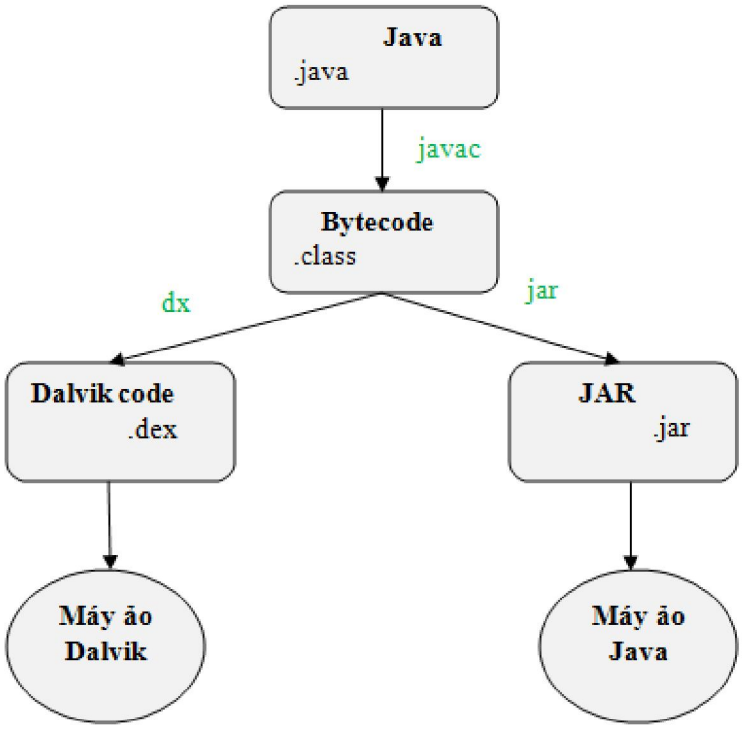
- Tầng **Libraries**: Android có hệ thống các thư viện C/C++ được sử dụng nhiều trong các thành phần khác nhau của hệ điều hành. Một số các thư viện C/C++ chính trong Android:
  - System C Library: một BSD (Berkely Software Distribution) được thừa kế từ các thư viện chuẩn C và được tinh chỉnh cho các thiết bị sử dụng trên nền Linux.
  - Media Library: thư viện hỗ trợ cho việc ghi âm, chơi các định dạng nhạc, phim và hiển thị các ảnh bao gồm các định dạng sau: MPEG4, H.264, MP3, AAC, ARM, JPG, PNG…
  - Surface Manager: Quản lý truy cập vào hệ thống hiển thị.
  - Live Webcore: Công cụ trình duyệt web.
  - SGL: Các hàm cơ bản về đồ họa 2 chiều.
  - 3D Library: Đồ họa 3 chiều.
  - Freetype: Biểu diễn các font và vectơ bitmap.
  - SQLite: Cơ sở dữ liệu.
- Tầng **Android Runtime**: Bao gồm một tập các thư viện lõi Java và máy ảo Dalvik. Máy ảo Dalvik thực thi các tập tin định dạng dex. Mỗi

ứng dụng được chạy trên một tiến trình riêng của máy ảo Dalvik. Trên cùng 1 thiết bị có thể chạy nhiều máy ảo Dalvik khác nhau một cách hiệu quả.

- Tầng **Linux Kernel:** Đây là tầng thấp nhất trong hệ điều hành Android, được xây dựng trên nhân của Linux 2.6 chứa các trình quản lý thiết bị như keypad, wifi, âm thanh, quản lý điện năng… và các dịch vụ của hệ thống như: an ninh, quản lý bộ nhớ, quản lý tiến trình, kết nối mạng. Tầng này đóng vai trò là tầng trung gian liên lạc giữa phần cứng và ngăn xếp phần mềm ở các tầng trên.

2.1.4. Máy ảo Dalvik

Dalvik là máy ảo để thực hiện các ứng dụng phần lớn viết bằng Java trên Android dưới dịnh dạng là tập tin (.dex). Về cơ bản có thể nhận thấy máy ảo Dalvik có phần giống với máy ảo Java trên Desktop, tuy nhiên có phần khác là khi ta viết các ứng dụng trên Java thì mã nguồn sẽ được chuyển thành mã bytecode. Tại đây, một công cụ có sẵn trên Android là "dx" sẽ chuyển dạng mã bytecode này thành dạng tập tin .dex (viết tắt là Dalvik Excutable) và được thực thi trên máy ảo Dalvik để chạy các ứng dụng Android.



Hình 2.3 Cơ chế hoạt động của máy ảo Dalvik và Java

2.1.5. Android software development kit (SDK)

Bộ phát triển ứng dụng cho Android hay còn gọi là Android SDK cung cấp cho các nhà phát triển phần mềm có thể lập trình, gỡ lỗi và kiểm thử ứng dụng được phát triển trên Android. Bộ SDK bao gồm:

- Thư viện lập trình Android (Android API): đây là phần cốt lõi của bộ phát triển Android, từ các thư viên lập trình Android API, Google đã xây dựng nên các ứng dụng có sẵn.
- Công cụ phát triển: cung cấp sẵn cho các nhà phát triển các công cụ để lập trình, biên dịch, sửa lỗi mã nguồn trong ứng dụng.
- Tài liệu: đây là phần hướng dẫn sử dụng các thư viện, lớp / hàm có sẵn trong môi trường lập trình Android. Ngoài ra còn giải thích về cơ chế hoạt động của các ứng dụng trong Android.

- Ứng dụng mẫu: Bộ Android SDK còn cung cấp các đoạn mã nguồn của chương trình có sẵn.
- Trình giả lập Android: Để cung cấp sự thuận tiện cho người phát triển, bộ Android SDK đã cung cấp cho người dùng sẵn trình giả lập Android mô phỏng môi trường làm việc y như trên thiết bị thật để thuận tiện cho người phát triển có thể chạy hoặc sửa lỗi các ứng dụng trên thiết giả lập này mà không cần phải có thiết bị thật.

Hình 2.4 Minh họa trình giả lập điện thoại Android

2.2. Native development kit (NDK)

2.2.1. Giới thiệu chung

Khi viết một ứng dụng Android ở tầng trên bằng ngôn ngữ Java mà ta có nhu cầu gọi lại các hàm hoặc thư viện ở tầng bên dưới (thường là các đoạn mã ở tầng dưới được viết bằng C/C++). Để Java có thể hiểu và truy xuất được các đoạn mã C/C++ thì ta cần một giao diện chung giữa 2 ngôn ngữ. Giao diện chung đó được gọi là Java Native Interface – JNI.

Native Development Kit – NDK là một bộ công cụ đi kèm với Android SDK giúp cho các nhà phát triển có thể viết hoặc nhúng các đoạn mã nguồn bằng C/C++ bên trong chương trình. Các ứng dụng Android hoạt động trên máy ảo Dalvik. Chính nhờ NDK mà các ứng dụng có thể gọi được các đoạn mã gốc – native code được sử dụng trong chương trình.

2.2.2. Các hỗ trợ của NDK

Bộ công cụ NDK cung cấp các hỗ trợ sau:

- Một tập hợp các công cụ và tập tin để phát sinh ra các thư viện mã từ C/C++.
- Cách thức nhúng các đoạn mã phát sinh từ C/C++ vào trong tập tin đóng gói ứng dụng (.apk) chạy được trên các thiết bị Android.
- Cung cấp một tập các header và thư viện sẽ được hỗ trợ ở tất cả các phiên bản Android từ 1.5 trở đi. Từ phiên bản 2.3 có hỗ trợ thêm viết Native Activity.
- Các tài liệu, mã nguồn mẫu và hướng dẫn.

2.2.3. Sử dụng NDK

Không phải lúc nào sử dụng NDK cũng có lợi cho chương trình. Vì sử dụng mã gốc (native code) trong chương trình không làm tăng hiệu năng thực thi mà chỉ làm tăng thêm sự phức tạp cho ứng dụng. Chỉ sử dụng mã gốc trong trường hợp cần thiết để làm giảm sự phức tạp cho chương trình.

Android Framework cung cấp 2 cách để sử dụng native code trong chương trình:

- Viết ứng dụng sử dụng Android Framework và sử dụng JNI để truy cập các hàm API được cung cấp trong bộ công cụ Android NDK. Ưu điểm của kỹ thuật này là chúng ta có thể tận dụng các lợi ích của Android Framework mà vẫn sử dụng được mã gốc khi cần thiết.
- Viết một native activity để hiện thực cài đặt chu trình của ứng dụng bằng mã gốc. Bộ công cụ Android SDK sẽ cung cấp lớp NativeActivity

là lớp tiện ích để hiện thực cái đặt vòng đời của ứng dụng thông qua các hàm (OnCreate, OnPause…).

2.2.4. Nội dung của bộ NDK

Bao gồm các công cụ và thư mục sau:

Công cụ phát triển: Bao gồm tập hợp các công cụ phát triển (trình biên dịch, trình liên kết – linker) để phát sinh ra mã nhị phân cho bộ vi xử lý ARM chạy trên các nền tảng Linux, OS X và Windows (sử dụng kèm với công cụ Cygwin). Các công cụ phát triển này còn cung cấp một tập hợp các hệ thống header dùng cho các hàm API gốc ổn định và được đảm bảo là sẽ hỗ trợ trong tất cả các phiên bản sau này của nền tảng Android:

- Libc (thư viện C) header.
- Libm (thư viện toán học) header.
- Giao diện JNI header.
- Libz (nén và giải nén) header.
- Liblog (dùng cho việc ghi log trên Android) header.
- OpenGL ES 1.1 và OpenGL ES 2.0 (thư viện đồ họa ba chiều) header.
- Libjnigraphics (truy cập vùng nhớ đệm trên các pixel) header.
- Các header hỗ trợ cho C++.
- OpenSL ES (thư viện âm thanh gốc).
- Các API hỗ trợ ứng dụng gốc trên Android.

Ngoài ra NDK còn cung cấp cho chúng ta một hệ thống biên dịch mã nguồn hiệu quả mà không cần phải có sự điều khiển chi tiết các công cụ / nền tảng / vi xử lý / ABI. Người dùng sẽ chỉ phải tạo ra các tập tin nhỏ để chỉ thị cho việc biên dịch mã nguồn sẽ được dùng trong chương trình. NDK sẽ dựa vào các tập tin biên dịch này để biên dịch mã nguồn để tạo ra thư viện liên kết động và đặt trực tiếp thư viện này trong dự án.

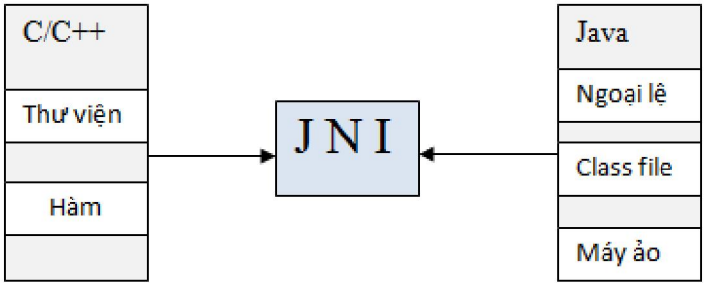
Bộ tài liệu: Bộ NDK còn chứa tập hợp nhiều tài liệu để mô tả các tính năng của NDK, cách thức sử dụng, viết tập tin biên dịch mã nguồn, cách

thức tạo thư viện liên kết động… Người dùng có thể tham khảo thêm trong thư mục <ndk>/docs.

Các ứng dụng mẫu: Cung cấp các ứng dụng được viết sẵn cho người dùng tham khảo.

2.2.5. Giới thiệu về JNI – Java native interface

Như chúng ta đã biết, muốn chạy được mã C/C++ trên Android thì chúng ta cần phải sử dụng JNI hoặc viết một NativeActivity. Và cách sử dụng JNI trên Android sẽ phổ biến hơn do tương thích với nhiều loại thiết bị. Trong luận văn, chúng em cũng sử dụng cách này để chạy mã nguồn C/C++ trên Android.



Hình 2.5 JNI đóng vai trò trung gian trong việc giao tiếp giữa C/C++ và Java

Sau đây là nguyên tắc hoạt động và cách thức viết một JNI trong chương trình [4]:

- Các phương thức được đánh dấu là JNI sẽ thêm từ khóa ***native*** ở đầu mỗi hàm.
- Các hàm này sẽ được cài đặt bằng C/C++ và được đặt trong thư mục JNI của project. Các hàm C/C++ được NDK biên dịch thành tập tin thư viện liên kết động **.so**.
- Để load thư viện liên kết động này thì trong chương trình Java sẽ gọi phương thức ***System.LoadLibrary(…)***.
- Sau đó, khi nào có lời gọi hàm trong chương trình thì máy ảo sẽ tìm kiếm các hàm này trong thư viện liên kết động và thực thi các phương thức được cài đặt bằng C/C++.

Ví dụ về chương trình hello-jni trong Android:

- Ta viết tập tin hello-jni.c để cài đặt hàm trả về 1 chuỗi (trong đó tên phương thức được đặt theo thứ tự sau: tên package_tên lớp_tên phương thức.

```
Jstring
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv*
env, jobject thiz )
{
return (*env)->NewStringUTF(env, "Hello from JNI!");
}
```

Sau đó tại thư mục JNI ta tạo tập tin Android.mk là tập tin chỉ thị để biên dịch và cấu hình mã nguồn C/C++ trong Android.

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

# Here we give our module name and source file(s)
LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c

include $(BUILD_SHARED_LIBRARY)
```

Hình 2.6 Nội dung tập tin cấu hình biên dịch trong JNI

Trong tập tin cấu hình Android.mk ta thấy 2 dòng quan trọng là LOCAL_MODULE và LOCAL_SRC_FILES. Dòng đầu là chỉ thị tên của thư viện sẽ được tạo ra khi gọi lệnh của NDK và dòng sau là chỉ thị các tập tin mã nguồn sẽ được biên dịch thành thư viện trong Android.

Từ Command Line gọi lệnh ndk-build để biên dịch ra thư viện liên kết động hello-jni.so.

Tại chương trình chính Java, ta load thư viện lên và gọi phương thức **native** của hàm trong thư viện:

```
static {
    System.loadLibrary("hello-jni");
     }
```

```
Public native String stringFromJNI();
```

Cuối cùng sử dụng phương thức bằng cách gọi hàm bình thường.

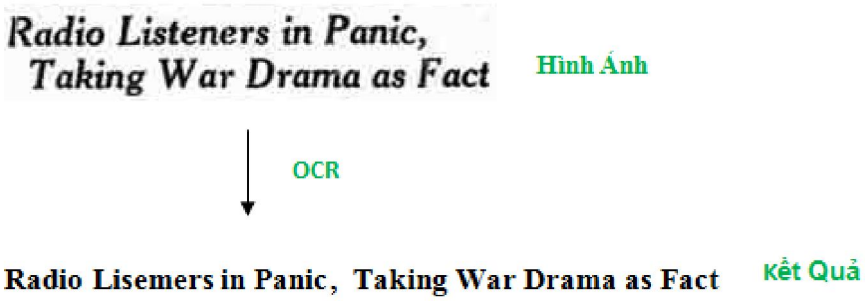
Chương 3 : NHẬN DẠNG KÝ TỰ QUANG HỌC

3.1. Giới thiệu chung

3.1.1. Sơ lược về nhận dạng ký tự quang học – OCR

Nhận dạng ký tự quang học (tên tiếng anh là Optical Character Recognition – OCR) là một quá trình thực hiện việc chuyển đổi từ dạng hình ảnh của chữ viết in hoặc các ký hiệu sang các dạng văn bản tài liệu hoặc thông tin có thể chỉnh sửa trên máy tính. Đầu vào của quá trình này là tập tin hình ảnh và đầu ra sẽ là các tập tin văn bản chứa nội dung là các chữ viết có trong hình ảnh đó. Nhận dạng ký tự quang học được hình thành từ các lĩnh vực nghiên cứu nhận dạng mẫu, trí tuệ nhân tạo và thị giác máy tính. Ngày nay kỹ thuật nhận dạng ký tự quang học đã được sử dụng rộng rãi và ứng dụng nhiều trong thực tế song song với việc nghiên cứu về lý thuyết để cải tiến kết quả nhận dạng.

Thông thường, các hệ thống nhận dạng ký tự quang học được sử dụng dưới dạng các phần mềm trong máy tính hoặc tích hợp trong máy in, máy quét để thực hiện việc nhận dạng ký tự. Ví dụ thường thấy nhất là quét các hình ảnh văn bản thành các văn bản tài liệu lưu trên máy tính.



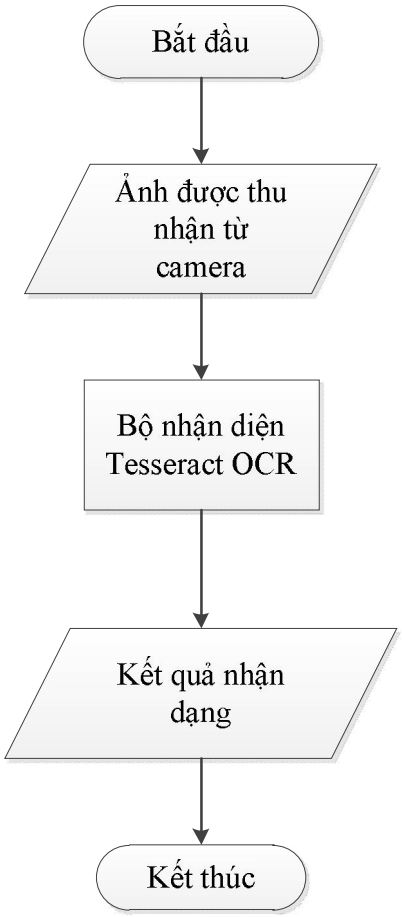
Hình 3.1 Quá trình thực hiện OCR

3.1.2. Các phương pháp áp dụng OCR trong luận văn

Luận văn tập trung vào việc sử dụng kỹ thuật nhận dạng ký tự quang học áp dụng trên điện thoại Android để thực hiện việc tra từ điển qua camera của điện

thoại. Sau đây là các phương pháp có thể áp dụng kỹ thuật nhận dạng ký tự quang học:

- **Nghiên cứu và tự xây dựng một bộ nhận dạng ký tự quang học**: Đây là cách khó khăn khi thực hiện vì hiện nay trên thế giới đã có nhiều hướng nghiên cứu về lĩnh vực này và cho ra đời nhiều phương pháp nhận dạng ký tự quang học. Tự viết lại bộ nhận dạng ký tự quang học sẽ tốn khá nhiều thời gian mà hiệu quả sẽ không được cao mà luận văn này chủ yếu tập trung vào việc sử dụng nhận dạng ký tự quang học để thực hiện tra từ nên cách này sẽ không khả thi và bản thân việc nghiên cứu các kỹ thuật nhận dạng ký tự quang học đã là một đề tài lớn nên chúng em sẽ không chọn phương pháp này để thực hiện nhận dạng ký tự quang học.
- **Sử dụng các bộ nhận dạng ký tự quang học trực tiếp trên web thông qua môi trường mạng**: Điện thoại sẽ gửi hình ảnh lên máy chủ web để máy chủ sẽ trực tiếp xử lý áp dụng các thuật toán nhận dạng ký tự quang học được cài đặt sẵn để xử lý, phân tích bức ảnh và gửi trả kết quả đã được nhận dạng về cho điện thoại. Cách này có ưu điểm là dễ thực hiện và độ chính xác có thể cao tuy nhiên khi sử dụng chương trình đòi hỏi người sử dụng phải cài đặt mạng điện thoại hoặc wifi trong máy để kết nối mạng internet cho việc truyền và nhận dữ liệu từ máy chủ trên web. Chưa kể đến việc xử lý và chờ kết quả từ máy chủ trên mạng sẽ khá lâu gây bất tiện cho người sử dụng chương trình. Nếu điện thoại không có kết nối mạng thì người dùng sẽ không thể sử dụng được tính năng nhận diện từ.
- **Sử dụng các bộ thư viện nhận dạng ký tự quang học có sẵn**: So với 2 cách được nêu ra ở trên thì cách này có ưu điểm là thực hiện không quá khó khăn, chỉ cần cài đặt và biên dịch để chạy trên môi trường Android vừa khắc phục được nhược điểm là ứng dụng không phụ thuộc vào môi trường mạng, có thể chạy độc lập, tiết kiệm nhiều thời gian vì việc xử lý trên khối nhận dạng được thực hiện hoàn toàn trực tiếp trên điện thoại.



Hình 3.2 Sơ đồ khối nhận diện ký tự quang học trong chương trình

3.1.3. So sánh các thư viện / công cụ nhận dạng ký tự quang học

Hiện nay trên thế giới đã có khá nhiều bộ thư viện nhận dạng ký tự quang học với độ chính xác khá cao. Sử dụng một trong các thư viện đó sẽ giúp chúng ta tiết kiệm khá nhiều công sức. Sau đây là một số bộ và phần mềm nhận dạng ký tự quang học miễn phí được sử dụng rộng rãi hiện nay:

- Tesseract OCR[2]: là bộ nhận dạng ký tự quang học thương mại ban đầu được phát triển tại công ty HP (Hewlett-Packard) trong khoảng 1985 – 1995 và được giải thưởng top 3 phần mềm nhận dạng ký tự quang học chính xác nhất trong hội nghị thường niên của UNLV (University of Nevada-Las Vegas). Sau đó bộ nhận dạng này được chuyển thành mã

[2] http://code.google.com/p/tesseract-ocr/

nguồn mở trên Google và tiếp tục được phát triển cho đến ngày nay với sự đóng góp của nhiều lập trình viên chuyên nghiệp. Trưởng bộ phận của dự án hiện nay là Ray Smith.

- GOCR[3]: Là một chương trình nhận dạng ký tự quang học được phát triển dưới dạng giấy phép công cộng GNU và được bắt đầu bởi Joerg Schulenberg vào năm 2000.
- FreeOCR[4]: Được xem là một trong các phần mềm nhận dạng ký tự quang học chính xác nhất vì sử dụng bộ engine Tesseract của HP. Ngoài ra, FreeOCR còn cung cấp dịch vụ nhận dạng ký tự quang học trực tuyến trên web.
- JavaOCR[5]: Là phần mềm nhận dạng ký tự quang học được viết hoàn toàn toàn bằng thư viện Java cho việc xử lý ảnh và nhận dạng ký tự. Ưu điểm của chương trình này là chiếm ít tài nguyên bộ nhớ, dễ thực hiện trên các môi trường di động hạn chế về bộ nhớ và chỉ sử dụng được ngôn ngữ Java.

3.1.4. Kết luận

Trong các thư viện nhận dạng ký tự quang học trên thì bộ nhận dạng Tesseract OCR nổi trội nhất với các ưu điểm sau:

- Có lịch sử phát triển lâu dài và mang độ chính xác cao ngay từ khi mới ra mắt.
- Khả năng mở rộng và tùy biến cao đồng thời được Google tài trợ và đông đảo các nhà phát triển tham gia đóng góp cho Tesseract.
- Phiên bản được cập nhật thường xuyên, hỗ trợ ngày càng nhiều ngôn ngữ, có khả năng huấn luyện trên các ngôn ngữ mới và nhiều loại font chữ khác nhau.
- Một số phần mềm OCR hiện nay đều sử dụng bộ nhận dạng này cho việc nhận dạng ký tự nên Tesseract đã trở nên phổ biến hơn, đồng thời

[3] http://jocr.sourceforge.net/
[4] http://www.free-ocr.com/
[5] http://sourceforge.net/projects/javaocr/

khả năng hỗ trợ trên nhiều môi trường, nền tảng khác nhau từ máy tính cho đến các thiết bị di động.

Chính vì các ưu điểm nêu trên mà trong luận văn này nhóm chúng em sẽ sử dụng Tesseract để thực hiện quá trình nhận dạng ký tự trong chương trình.

3.2. Giới thiệu về bộ nhận dạng ký tự quang học Tesseract

3.2.1. Lịch sử

Tesseract [5] là một phần mềm mã nguồn mở và ban đầu nó được nghiên cứu và phát triển tại hãng Hewlett Packet (HP) trong khoảng từ năm 1984 đến 1994. Vào năm 1995, Tesseract nằm trong nhóm ba bộ nhận dạng OCR đứng đầu về độ chính xác khi tham gia trong hội nghị thường niên của tổ chức UNLV.

Lúc mới khởi động thì Tesseract là một dự án nghiên cứu tiến sĩ tại phòng thí nghiệm HP ở Bristol và đã được tích hợp vào trong các dòng máy quét dạng phẳng của hãng dưới dạng các add-on phần cứng hoặc phần mềm. Nhưng thực tế dự án này đã thất bại ngay từ trong trứng nước vì nó chỉ làm việc hiệu quả trên các tài liệu in có chất lượng tốt.

Sau đó, dự án này cùng với sự cộng tác của bộ phận máy quét HP ở bang Colorado đã đạt được một bước tiến quan trọng về độ chuẩn xác khi nhận dạng và vượt lên nhiều bộ nhận dạng OCR thời đó nhưng dự án đã không thể trở thành sản phẩm hoàn chỉnh vì độ cồng kềnh và phức tạp. Sau đó, dự án được đưa về phòng thí nghiệm của HP để nghiên cứu về cách thức nén và tối ưu mã nguồn. Dự án tập trung cải thiện hiệu năng làm việc của Tesseract dựa trên độ chính xác đã có. Dự án này được hoàn tất vào cuối năm 1994 và sau đó vào năm 1995 bộ Tesseract được gửi đi tham dự hội nghị UNLV thường niên về độ chính xác của OCR, vượt trội hơn hẳn so với các phần mềm OCR lúc bấy giờ. Tuy nhiên, Tesseract đã không thể trở thành một sản phẩm thương mại hoàn chỉnh được và vào năm 2005, HP đã chuyển Tesseract sang mã nguồn mở và được hãng Google tài trợ. Tesseract cho đến nay vẫn được nhiều nhà phát triển cộng tác và tiếp tục hoàn thiện. Phiên bản mới nhất của bộ nhận dạng Tesseract là phiên bản 3.0.1.

Bảng 3.1 So sánh phần mềm thương mại và Tesseract

Phần mềm thương mại	Bộ nhận dạng Tesseract
Hỗ trợ hơn 100 ngôn ngữ	Hỗ trợ trên 40 ngôn ngữ và đang tăng dần
Có giao diện đồ họa	Không hỗ trợ giao diện đồ họa (dùng Command Line để gõ lệnh)
Hầu hết chỉ hỗ trợ trên nền tảng Windows	Hỗ trợ trên Windows, Linux, Mac OS
Độ chính xác cao mới đây	Độ chính xác cao từ năm 1995
Chi phí khá cao 130$ - 500 $	Hoàn toàn miễn phí (mã nguồn mở)

Vì Tesseract hiện nay là bộ thư viện mã nguồn mở hoàn toàn miễn phí nên trên thế giới đã có nhiều phần mềm nhận dạng ký tự quang học ra đời dựa trên bộ Tesseract với giao diện và các tính năng dễ sử dụng hơn so với giao diện đơn giản của Tesseract ban đầu như: VietOCR[6] cho nhận dạng tiếng Việt, Tessenet2[7] bộ nhận diện Tesseract trên nền .Net của Microsoft, giao diện Java (Java GUI frontend) cho Tesseract…

Bảng 3.2 Độ chính xác của Tesseract trên một số ngôn ngữ

Ngôn ngữ	Tổng số ký tự (triệu)	Tổng số từ (triệu)	Lỗi ký tự (%)	Lỗi từ (%)
Tiếng Anh	39	4	0.5	3.72
Tiếng Nga	213	26	0.75	5.78
Tiếng Hoa giản thể	0.25	không xác định	3.77	không xác định
Tiếng Hindi	1.4	0.33	15.41	69.44

[6] http://vietocr.sourceforge.net/usage_vi.html
[7] http://www.pixel-technology.com/freeware/tessnet2/

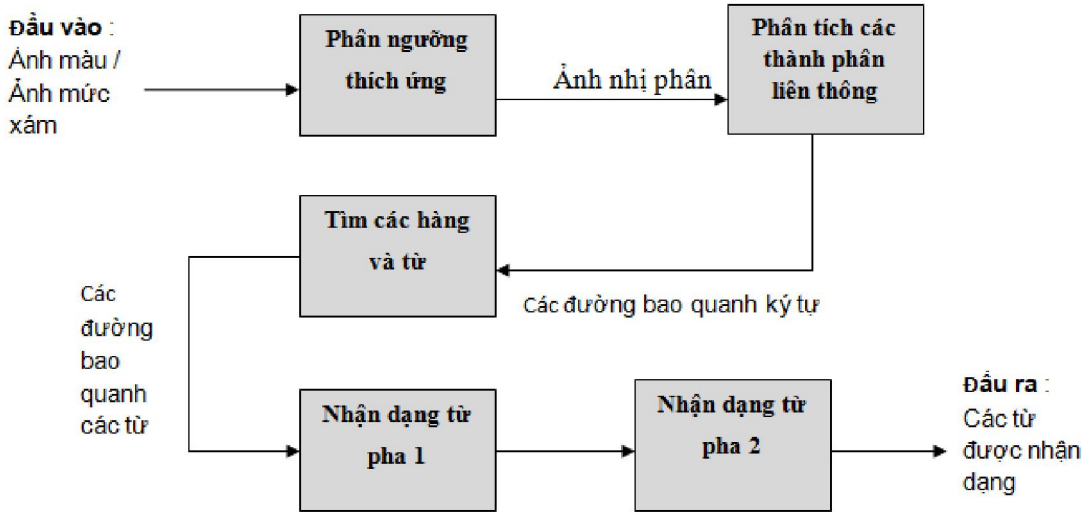
3.2.2. Kiến trúc hoạt động

Đầu tiên, bộ nhận diện Tesseract sẽ nhận đầu vào là ảnh màu hoặc ảnh mức xám. Ảnh này sẽ được chuyển đến bộ phận phân tích ngưỡng thích ứng (**adaptive thresholding**) để cho ra ảnh nhị phân. Vì trước kia HP cũng đã phát triển bộ phận phân tích bố cục trang nên Tesseract không cần phải có thành phần đó và được thừa hưởng từ HP. Vì thế mà Tesseract nhận đầu vào là một ảnh nhị phân với các vùng đa giác tùy chọn đã được xác định.

Ban đầu, Tesseract được thiết kế làm việc trên ảnh nhị phân sau đó chương trình được cải tiến để có thể nhận dạng cả ảnh màu và ảnh mức xám. Chính vì thế mà cần bộ phận phân tích ngưỡng thích ứng để chuyển đổi ảnh màu / ảnh mức xám sang ảnh nhị phân.

Sau đó quá trình nhận dạng sẽ được thực hiện tuần tự theo từng bước.

- Bước đầu tiên là phân tích các thành phần liên thông. Kết quả của bước này sẽ là tạo ra các đường bao quanh các ký tự.
- Bước thứ hai là tìm hàng và tìm từ, kết quả của bước này cũng giống như bước trên sẽ tạo ra các vùng bao quanh các hàng chữ và ký tự chứa trong vùng văn bản.
- Bước tiếp theo sẽ là nhận dạng từ. Công đoạn nhận dạng từ sẽ được xử lý qua 2 giai đoạn. Giai đoạn đầu sẽ là nhận dạng các từ theo lượt. Các từ thỏa yêu cầu trong giai đoạn này sẽ được chuyển sang bộ phân loại thích ứng [5] (**adaptive classifier**) để làm dữ liệu huấn luyện. Chính nhờ đó mà bộ phân loại thích ứng sẽ có khả năng nhận diện được chính xác hơn ở phần sau của trang. Sau khi bộ phân loại thích ứng đã học được các thông tin có ích từ giai đoạn đầu khi nhận dạng phần trên của trang thì giai đoạn thứ 2 của việc nhận dạng sẽ được thực hiện. Giai đoạn này sẽ quét hết toàn bộ trang, các từ không được nhận diện chính xác ở giai đoạn đầu sẽ được nhận diện lại lần nữa. Cuối cùng bộ nhận diện sẽ tổng hợp lại các thông tin ở trên và cho ra kết quả nhận diện hoàn chỉnh.



Hình 3.3 Kiến trúc tổng thể của Tesseract [2]

3.2.3. Cài đặt và sử dụng thư viện Tesseract trên Android

Thư viện mã nguồn của bộ Tesseract được viết bằng C/C++ chuẩn chạy trên các nền tảng Windows và Linux nên để có thể chạy được bộ thư viện trên nền tảng Android ta cần chuyển mã nguồn sang hệ điều hành này. Có 2 cách để thực hiện việc chuyển đổi mã nguồn chạy trên Android:

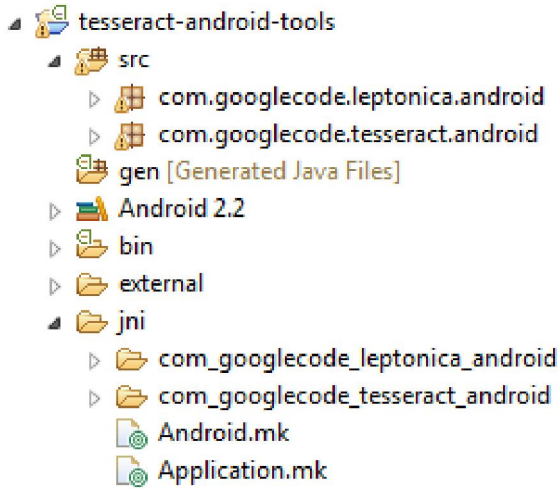
- Cách thứ nhất là viết lại mã nguồn Tesseract bằng thư viện lập trình Java vì các ứng dụng trên Android được viết bằng ngôn ngữ này. Ưu điểm của cách này là nếu thực hiện được thì chương trình sẽ hoạt động một cách hiệu quả vì mã nguồn Java hoạt động tối ưu trên hệ điều hành này. Tuy nhiên đối với người lập trình mà nói thì sẽ khó thực hiện được vì phải viết lại toàn bộ chương trình từ đầu, tiêu tốn rất nhiều thời gian mà sẽ không đạt hiệu quả cao.
- Cách thứ hai là nhờ vào cách thức sử dụng JNI trên Android để có thể sử dụng lại các hàm thư viện C/C++ trên bộ Tesseract. Cách này mang ưu điểm là sẽ dễ thực hiện hơn đối với người lập trình, tiết kiệm được nhiều thời gian, công sức và chương trình hoạt động theo đúng yêu cầu đề ra. Có chăng nhược điểm chỉ là chạy chậm hơn một chút so với cách trên vì Android phải thực thi mã C/C++ thông qua trung gian là JNI.

Dựa vào các ưu nhược điểm của từng phương pháp trên thì trong luận văn này nhóm chúng em đã chọn cách 2 để có thể sử dụng được thư viên Tesseract trên nền tảng Android.

Quá trình chuyển mã nguồn Tesseract để thực thi được trên Android:

- Để có thể chạy được mã nguồn Tesseract thì đầu tiên ta cần tải về chương trình NDK sau đó giải nén thư mục ra và chỉnh biến môi trường trỏ đến đường dẫn thư mục NDK vừa giải nén.
- Tải về project **tesseract-android-tools**[8], đây là một công cụ mã nguồn mở trên Google được viết trong trình biên dịch Eclipse cung cấp một tập các hàm API và các tập tin cấu hình để biên dịch thư viện Tesseract và thư viện xử lý ảnh Leptonica.
- Trình biên dịch Eclipse dùng để viết mã nguồn đồng thời được cài đặt sẵn các plug-in Android bao gồm bộ công cụ Android SDK với các phiên bản API thích hợp để phát triển và biên dịch chương trình trực tiếp trên Eclipse.

Sau quá trình chuẩn bị các công cụ cần thiết cho việc biên dịch mã nguồn Tesseract trên Android thì ta cần import project **tesseract-android-tools** vào trong Eclipse ta được như sau:



Hình 3.4 Minh họa cấu trúc của project tesseract-android-tools

[8] http://code.google.com/p/tesseract-android-tools/

Project này thực chất là 1 project trong Android có sử dụng JNI và NDK để biên dịch và chạy mã nguồn C/C++ trên Android. Ta để ý trong thư mục JNI là thư mục chứa các mã nguồn C/C++ và thực thi quá trình trong JNI. Tập tin Android.mk là tập tin để chỉ thị cho việc biên dịch các hàm và thư viên C/C++. Nhiệm vụ của ta là cần tinh chỉnh lại tập tin này để có thể biên dịch được thư viện trên.

Trước khi thực thi lệnh của NDK để chạy thư viện C/C++ thì ta cần chuẩn bị các mã nguồn của các thư viện sau:

- Mã nguồn của **Tesseract** phiên bản 3.01 (quan trọng nhất).
- Hai thư viện xử lý ảnh **leptonica-1.68** và **libjpeg**.

Tạo thư mục tên external đặt trong project **tesseract-android-tools** chứa mã nguồn của 3 thư viện này. Mở tập tin Android.mk trong tất các thư mục con của thư mục JNI ta sẽ thấy các chỉ thị để biên dịch các tập tin mã nguồn của các thư viện này.

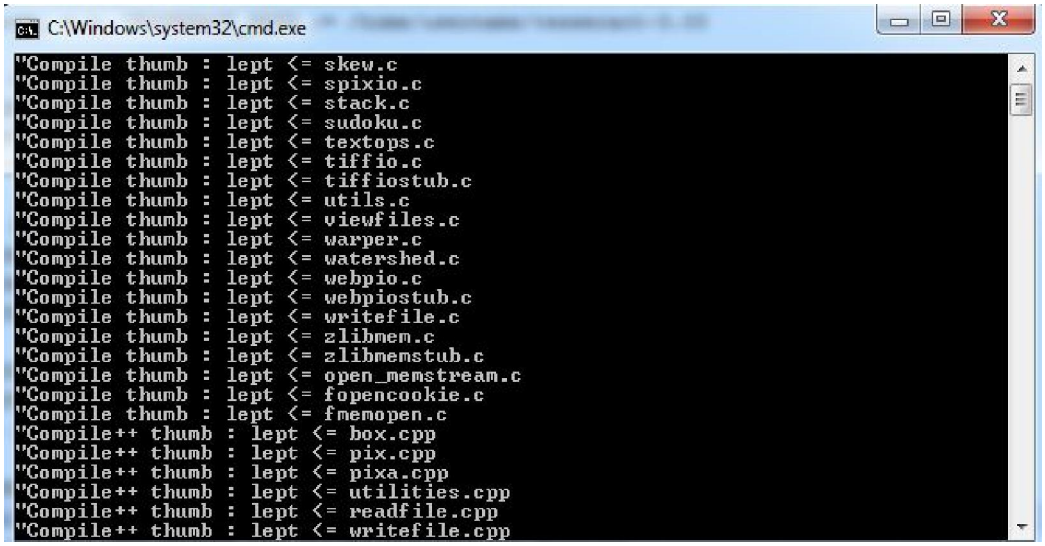
```
LOCAL_MODULE := libtess

# scrollview_lib

LOCAL_SRC_FILES := \
  $(TESSERACT_PATH)/viewer/scrollview.cpp \
  $(TESSERACT_PATH)/viewer/svutil.cpp \
  $(TESSERACT_PATH)/viewer/svmnode.cpp \
  $(TESSERACT_PATH)/viewer/svpaint.cpp

LOCAL_C_INCLUDES := \
  $(LEPTONICA_PATH)/src

LOCAL_CFLAGS := \
  -DHAVE_LIBLEPT

# tesseract_ccutil

LOCAL_SRC_FILES += \
  $(TESSERACT_PATH)/ccutil/ambigs.cpp \
  $(TESSERACT_PATH)/ccutil/basedir.cpp \
  $(TESSERACT_PATH)/ccutil/bits16.cpp \
  $(TESSERACT_PATH)/ccutil/boxread.cpp \
  $(TESSERACT_PATH)/ccutil/clst.cpp \
```

Hình 3.5 Minh họa một phần các chỉ thị để biên dịch mã nguồn thư viện C/C++ trong tập tin Android.mk

Ta có thể thực hiện quá trình biên dịch 3 thư viện trên bằng cách gõ lệnh NDK trên Windows / Linux / Mac OS đều thực thi cùng quá trình như nhau. Trong luận

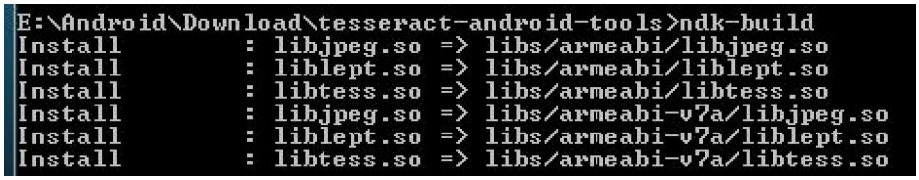
văn này nhóm chúng em sử dụng Command Line (CMD) trên Windows để thực thi quá trình biên dịch mã nguồn của Tesseract.

Trên Command Line của Windows, trỏ đường dẫn hiện thời đến thư mục project **tesseract-android-tools** trong Eclipse và gõ lệnh **ndk-build** để thực thi quá trình biên dịch mã nguồn của các thư viện.



Hình 3.6 Quá trình sử dụng NDK để biên dịch thư viện C/C++ trên Android

Trong quá trình biên dịch mã nguồn sẽ gặp nhiều thông báo lỗi nên ta cần sửa lại các tập tin cấu hình. Vì quá trình để biên dịch cũng chiếm khá nhiều thời gian của luận án nên chúng em không thể nêu ra chi tiết hết được. Cuối cùng sau khi quá trình biên dịch mã nguồn hoàn toàn thành công, sẽ tạo được 3 tập tin là **lipjpeg.so**, **liplept.so**, **libtess.so**. Đây là các tập tin thư viện trên Android và từ đây chúng ta có thể gọi trực tiếp các hàm C/C++ trong Tesseract thông qua các tập tin thư viện này.



Hình 3.7 Quá trình biên dịch mã nguồn thư viện Tesseract thành công trên Android

Sau khi hoàn tất xong, để gọi đến các hàm có sẵn trong thư viện Tesseract ta chú ý đến lớp TessBaseAPI trong thư mục **src** của project. Về bản chất, lớp này

đóng vai trò là lớp thực thi lại các hàm trên Tesseract, các hàm trong lớp này được cài đặt bằng Java và trong thân các hàm này gọi lại các hàm C/C++ bên dưới các tập tin thư viện.

Trong lớp TessBaseAPI ta lưu ý đến các hàm chính sau:

```
public boolean init(String datapath, String language)
public void setImage(Bitmap bmp)
public String getUTF8Text()
```

Hàm **init** là hàm dùng để khởi tạo các dữ liệu ban đầu cho Tesseract nhận vào 2 tham số: tham số datapath là chuỗi đường dẫn chỉ đến tập tin dữ liệu của bộ Tesseract OCR chứa trong thẻ nhớ SD Card của điện thoại, tham số thứ 2 chỉ định ngôn ngữ sẽ được nhận dạng.

Hàm **SetImage** để nhận vào hình ảnh dạng bitmap, hình ảnh này sẽ được nhận dạng sau đó thông qua hàm **getUTF8** và kết quả trả về của hàm này là một chuỗi có trong hình đó.

3.2.4. Huấn luyện dữ liệu trên Tesseract

Tesseract ban đầu được thiết kế để nhận dạng các từ tiếng Anh trên ngôn ngữ hệ Latinh. Sau này, nhờ sự cố gắng của nhiều nhà phát triển mà các phiên bản của Tesseract đã có thể nhận diện các ngôn ngữ khác ngoài hệ Latinh như tiếng Trung, tiếng Nhật và tương thích với các ký tự trong bảng mã UTF-8. Việc nhận dạng các ngôn ngữ mới trên Tesseract có thể thực hiện được nhờ vào việc huấn luyện dữ liệu. Từ phiên bản 3.0 trở đi, Tesseract đã có thể hỗ trợ thêm nhiều dạng ngôn ngữ mới và mở rộng thêm việc huấn luyện theo font chữ. Bởi vì ban đầu, bộ Tesseract được huấn luyện để nhận diện từ chính xác nhất trên một số loại font mặc định, nếu sử dụng các font chữ khác để nhận diện thì có thể kết quả sẽ không có độ chính xác cao khi làm việc với các loại font được cài đặt sẵn trong dữ liệu huấn luyện. Để thực hiện quá trình huấn luyện thì ta phải sử dụng công cụ có sẵn của Tesseract. Mặc định trong luận văn này, sử dụng công cụ Tesseract 3.01 cho việc thực hiện huấn luyện ngôn ngữ và font mới.

Để huấn luyện dữ liệu trên Tesseract (hoặc ngôn ngữ mới) thì ta cần một tập các tập tin dữ liệu chứa trong thư mục **tessdata**, sau đó kết hợp các tập tin này thành tập tin duy nhất. Các tập tin có trong thư mục **tessdata** có quy tắc đặt tên theo dạng: **tên_ngôn_ngữ.tên_tập tin**. Ví dụ các tập tin cần thiết khi thực hiện việc huấn luyện tiếng Anh:

- *tessdata/eng.config.*
- *tessdata/eng.unicharset:* Tập ký tự của ngôn ngữ huấn luyện.
- *tessdata/eng.unicharambigs.*
- *tessdata/eng.inttemp:* Danh mục cho tập hợp các ký tự.
- *tessdata/eng.pffmtable:* Tập tin dạng hộp – sử dụng để xác định ký tự có trong tập tin huấn luyện.
- *tessdata/eng.normproto:* Như tập tin *pffmtable.*
- *tessdata/eng.punc-dawg.*
- *tessdata/eng.number-dawg.*
- *tessdata/eng.freq-dawg:* Danh sách các từ tổng quát.
- *tessdata/eng.word-dawg:* Danh sách các từ thông thường.
- *tessdata/eng.user-word:* Danh sách từ của người dùng (tùy chọn có thể có hoặc không).

Bước cuối cùng sẽ tổng hợp dữ liệu từ bước trên và phát sinh ra tập tin dữ liệu duy nhất có dạng:

- *tessdata/eng.traineddata.*

Các tập tin cần thiết cho việc huấn luyện dữ liệu sẽ được phát sinh khi ta sử dụng công cụ có sẵn để qua quá trình huấn luyện.

3.2.5. Quá trình huấn luyện ngôn ngữ và font mới

Để trải qua quá trình huấn luyện ngôn ngữ hoặc loại font mới trên Tesseract ta cần thực hiện thông qua các giai đoạn sau:

- Phát sinh các tập tin hình ảnh cho việc huấn luyện:

Đây là bước đầu tiên nhằm xác định tập ký tự sẽ được sử dụng trong việc huấn luyện. Trước hết ta cần chuẩn bị sẵn một tập tin văn bản chứa các dữ liệu huấn luyện (trường hợp cụ thể là một đoạn văn bản). Việc tạo ra tập tin huấn luyện cần theo các quy tắc sau:

- Bảo đảm số lần xuất hiện ít nhất của các ký tự trong mẫu từ khoảng 5 đến 10 lần cho một ký tự.
- Nên có nhiều mẫu cho các từ xuất hiện thường xuyên, ít nhất là 20 lần.
- Các dữ liệu huấn luyện nên được chia theo kiểu font, mỗi tập tin huấn luyện chỉ nên chứa 1 loại font nhưng có thế huấn luyện nhiều loại font cho nhiều tập tin. Không nên kết hợp nhiều loại font trong riêng một tập tin huấn luyện.

Sau khi đã chuẩn bị mẫu văn bản dùng cho việc huấn luyện thì ta cần phát sinh ra ảnh từ tập tin đó. Dùng các phần mềm để chuyển tập tin mẫu văn bản sang dạng tập tin ảnh hoặc in mẫu văn bản sau đó quét thành tập tin hình ảnh dạng **.tif** với độ phân giải là 300dpi. Tập tin cuối cùng trước khi thực hiện việc huấn luyện là tập tin ảnh dạng **.tif**.

Tạo các tập tin dạng hộp **.box**:

Một dạng tập tin để Tesseract có thể huấn luyện dựa trên các dữ liệu hình ảnh đã có bước đầu là tập tin dạng hộp – box. Tập tin dạng hộp là tập tin văn bản chứa 1 dãy các ký tự tuần tự từ đầu đến cuối trong tập tin hình ảnh, mỗi hàng chứa thông tin của 1 ký tự, tọa độ và đường bao quanh ký tự đó trong tập tin ảnh.

Để tạo ra tập tin dạng hộp ta sẽ dùng cách gõ lệnh (trên Windows là CMD và Linux là Terminal) sau (yêu cầu người dùng phải cài đặt công cụ Tesseract để có thể chạy được các lệnh này):

```
Tesseract [lang].[fontname].exp[num].tif
[lang].[fontname].exp[num] batch.nochop makebox
```

Sau khi thực hiện câu lệnh trên thì ta sẽ tạo ra được các tập tin dạng hộp .box.

Hình 3.8 Cấu trúc tập tin dạng hộp

- Chạy công cụ Tesseract trên máy tính để thực hiện việc huấn luyện dữ liệu. Sau khi được tập tin .box thì chúng ta cần 1 trình chỉnh sửa tập tin dạng hộp để kiểm tra lại và chỉnh sửa lại các thông số của từng ký tự cho khớp với văn bản ban đầu trong tập tin ảnh huấn luyện. Ở đây nhóm em dùng phần mềm **jTextBoxEditor** để chỉnh sửa trực tiếp tập tin dạng hộp.
- Sau khi kiểm tra và chỉnh sửa lại các ký tự cho chính xác trong tập tin dạng hộp thì thực hiện lệnh tiếp theo:

```
Tesseract      [lang].[fontname].exp[num].tif
[lang].[fontname].exp[num]  nobatch box.train.stderr
```

Nếu thành công thì tại giai đoạn này, Tesseract sẽ phát sinh ra tập tin .tr

- Ước lượng tập ký tự của ngôn ngữ cần huấn luyện: Tesseract cần biết hết các tập ký tự có thể xuất hiện trong dữ liệu. Ta dùng lệnh sau:

```
unicharset_extractor *.box
```

Sau khi thực hiện, tập tin ***unicharset*** sẽ được tạo ra.

- Xác định kiểu font trong dữ liệu (từ phiên bản 3.0.1 trở đi):

Đây là tính năng mới chỉ có từ phiên bản Tesseract 3.0.1 trở đi. Với tính năng này người dùng có thể huấn luyện dữ liệu với nhiều loại font khác

nhau thay vì chỉ có thể dùng các font mặc định sẵn ở các phiên bản trước. Ta cần tạo tập tin font_properties để quy định thông số các kiểu font ta đã sử dụng trong các mẫu văn bản huấn luyện.

Cấu trúc của tập tin font_properties là mỗi hàng chứa tên 1 loại font huấn luyện và các đặc tính của font đó: <tên loại font><in nghiêng><in đậm><bình thường><in hoa><fraktur> (đánh dấu có thuộc tính bằng bit 1 hoặc không có dùng bit 0).

Ví dụ cấu trúc tập tin font_properties với dữ liệu huấn luyện là tiếng Anh:

```
Arial 0 0 0 0 0
arialbd 0 1 0 0 0
arialbi 1 1 0 0 0
ariali 1 0 0 0 0
```

- Gom nhóm dữ liệu:

Tại giai đoạn này thì các đường nét khung của ký tự đã được rút trích ra và chúng ta cần gom nhóm lại các dữ liệu ban đầu để tạo ra mẫu thử (prototype). Hình dạng, đường nét của các ký tự sẽ được gom nhóm lại nhờ vào chương trình **mftraining** và **cntraining** có sẵn trong công cụ Tesseract:

```
mftraining -F font_properties -U unicharset -O
lang.unicharset *.tr
```

Với lệnh **mftraining** sẽ tạo ra tập tin dữ liệu: **inttemp** (chứa hình dạng mẫu), **pffmtable** và **Microfeat** nhưng ít khi sử dụng).

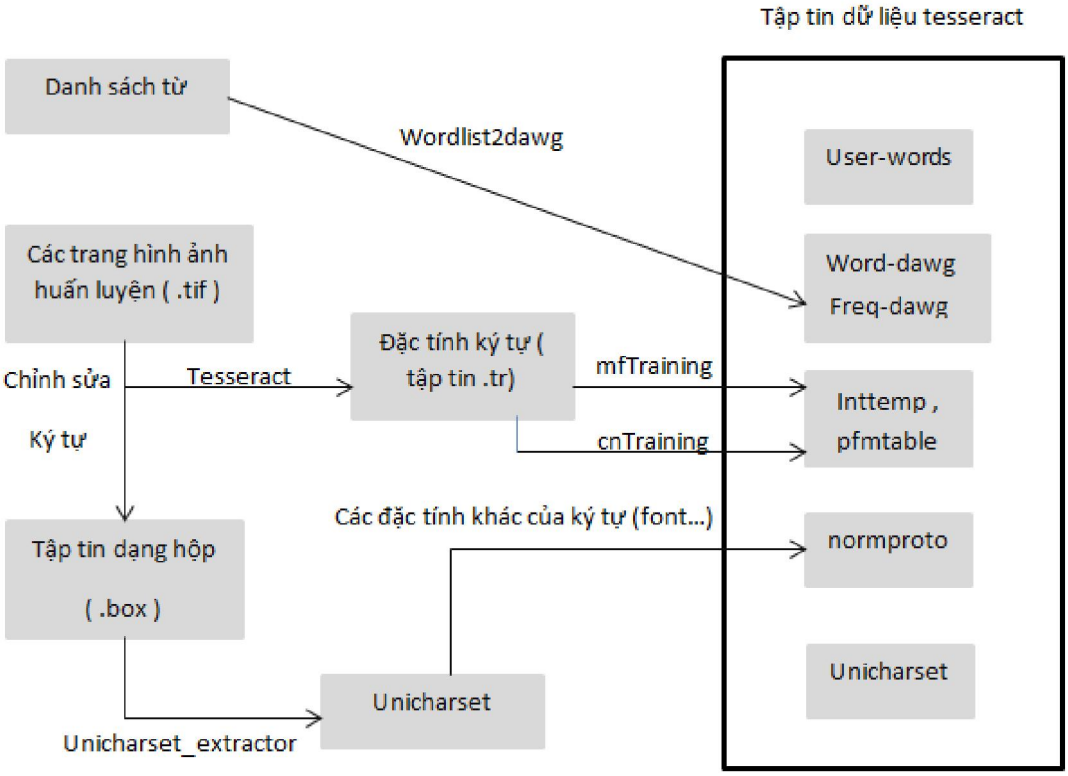
Cuối cùng dùng công cụ cntraining sẽ tạo ra tập tin dữ liệu **normproto**.

- Tạo tập tin **unicharambigs**.
- Kết hợp các tập tin lại tạo thành tập tin huấn luyện dữ liệu: Cuối cùng sau khi đã có đủ các tập tin huấn luyện cần thiết (inttemp, pffmtable, normproto, Microfeat) thì ta đổi tên các tập tin lại cho đúng dạng với tiền

tố **lang.** trước tên tập tin với lang là 3 ký tự đại diện cho ngôn ngữ huấn luyện theo chuẩn ISO 639-2[9]. Thực hiện lệnh sau:

```
Combine_tessdata lang.
```

Kết quả là tạo ra tập tin **lang.trainedata**. Bỏ tập tin này vào thực mục tessdata của Tesseract thì Tesseract đã có thể nhận diện được ngôn ngữ hoặc font chữ mới (theo lý thuyết).



Hình 3.9 Quá trình huấn luyện dữ liệu trên Tesseract

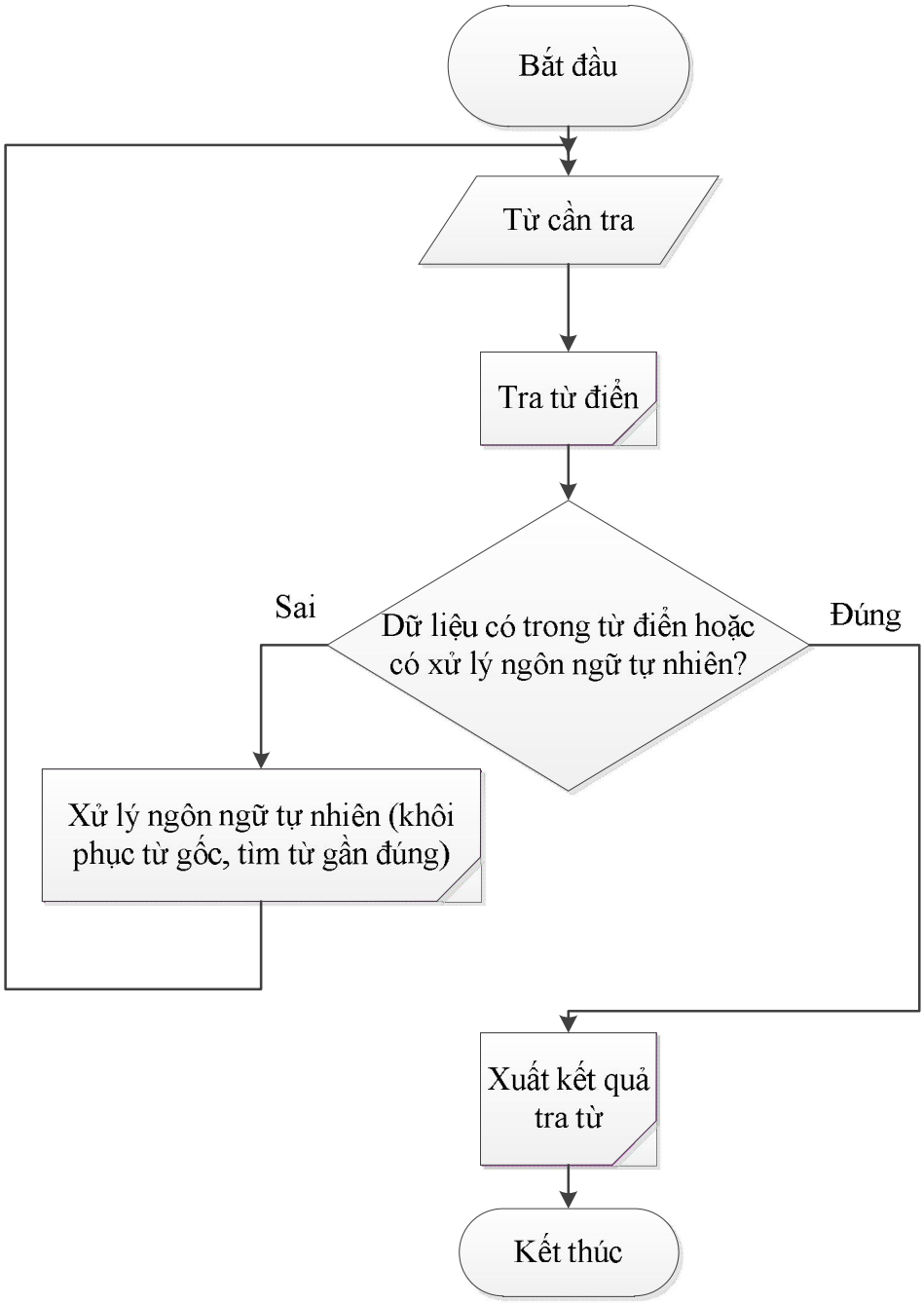
[9] http://en.wikipedia.org/wiki/List_of_ISO_639-2_codes

Chương 4 : TRA TỪ ĐIỂN ANH-VIỆT

4.1. Tổng quan

Trong khối này dữ liệu đầu vào là kết quả của khối xử lý dữ liệu ký tự quang học từ OCR và cho đầu ra là kết quả tra cứu từ điển. (Sơ đồ hình 4.1)

Nếu từ ban đầu có trong dữ liệu từ điển thì trả về kết quả tra từ ngay lập tức, ngược lại từ đó có thể sai do hai trường hợp do đó phải qua một bước kiểm tra xử lý ngôn ngữ tự nhiên. Trường hợp 1: từ chụp được không chính xác do kết quả nhận dạng sai, để giải quyết trường hợp này, ứng dụng sử dụng thuật toán tìm từ gần đúng để liệt kê danh sách các từ liên quan đến từ sai vừa chụp. Trường hợp 2: từ chụp do dạng biến thể của từ vựng, do có thêm các tiền tố, hậu tố nên trong từ điển không tồn tại dữ liệu, để giải quyết trường hợp này ứng dụng dùng thuật toán khôi phục từ gốc để trả về nguyên mẫu. Trường hợp ngược lại từ đó hoàn toàn không có trong từ điển thì ứng dụng thông báo không có kết quả thì ứng dụng tìm các từ tương tự để gợi ý cho người dùng chọn lựa.



Hình 4.1 Sơ đồ thuật toán tra từ điển và xử lý ngôn ngữ tự nhiên

Khi xây dựng ứng dụng từ điển trên điện thoại thì có hai điều khó khăn cần quan tâm là tốc độ xử lý và bộ nhớ. Hai vấn đề này rất quan trọng trong mối quan hệ giữa thiết bị đi động và ứng dụng. Nếu muốn tốc độ xử lý nhanh thì tốn bộ nhớ và ngược lại ứng dụng cần nhiều bộ nhớ thì ảnh hưởng tốc độ xử lý. Môi trường

trên di động thường giới hạn cả về bộ nhớ lẫn tốc độ xử lý. Do đó ta phải giải quyết hai vấn đề này cho thỏa mản yêu cầu ứng dụng.

-Vấn đề bộ nhớ: để giải quyết vấn đề này phải tăng dung lượng bộ nhớ của thẻ nhớ trên thiết bị di động. Hiện tại các dung lượng thẻ nhớ không còn là vấn đề khó khăn nên việc này được giải quyết.

-Tốc độ xử lý: bộ vi xử lý của thiết bị di động thì khó có thể nâng cấp được do đó chúng ta phải tổ chức cấu trúc dữ liệu từ điển để tăng tốc độ tra từ nhanh hơn.

Như vậy ứng dụng không những giải quyết các vấn đề về xử lý các ngôn ngữ tự nhiên mà còn tổ chức cấu trúc dữ liệu từ điển hỗ trợ tìm kiếm nhanh.

4.2. Khôi phục từ gốc (Stemming)

Tiếng Anh là ngôn ngữ thuộc loại hình ngôn ngữ hòa kết (flexional). Các hình vị trong ngôn ngữ hòa kết thường không đứng một mình mà đi kèm phụ tố, mỗi phụ tố có thể mang đồng thời nhiều ý nghĩa, hoặc ngược lại một ý nghĩa có thể biểu diễn bằng nhiều phụ tố. Trong tiếng Anh các phụ tố có thể tạo ra các **dẫn xuất hoặc biến cách** khác nhau.

Một từ trong văn bản tiếng Anh có thể có nhiều thể hiện khác nhau dưới nhiều dạng ngữ pháp khác nhau, tuy nhiên chúng cùng mang một nội dung ngữ nghĩa. Nên chúng được xem xét là một. Ví dụ: look, looks, looking, looked,… Các từ dạng này thường là danh từ số nhiều, động từ ở ngôi thứ ba số ít, động từ ở dạng thêm -ing hoặc dạng quá khứ, quá khứ phân từ. Do đó ứng dụng phải khôi phục từ gốc các dạng này. Từ gốc là một phần của từ sau khi loại bỏ các phụ tố. Phụ tố có thể là tiền tố hoặc hậu tố. Ví dụ các tiền tố như: dis-, un-, muti-… các hậu tố như: -ly, -ment, -tion, -logy… Với mỗi phụ tố khác nhau sẽ tạo ra dẫn xuất hoặc biến cách khác nhau và có cách xử lý cụ thể cho từng trường hợp.

Đối với tiền tố tạo ra dẫn xuất của từ, thì từ đó sẽ mang ngữ nghĩa khác, do đó chúng ta không cần phải thực hiện khôi phục từ gốc. Ví dụ: like và unlike là khác nhau.

Đối với hậu tố có hai trường hợp: tạo ra dẫn xuất hoặc tạo ra biến cách. Hậu tố tạo ra dẫn xuất sẽ có ngữ nghĩa khác nhau, hoặc từ loại khác nhau. Ví dụ: apply,

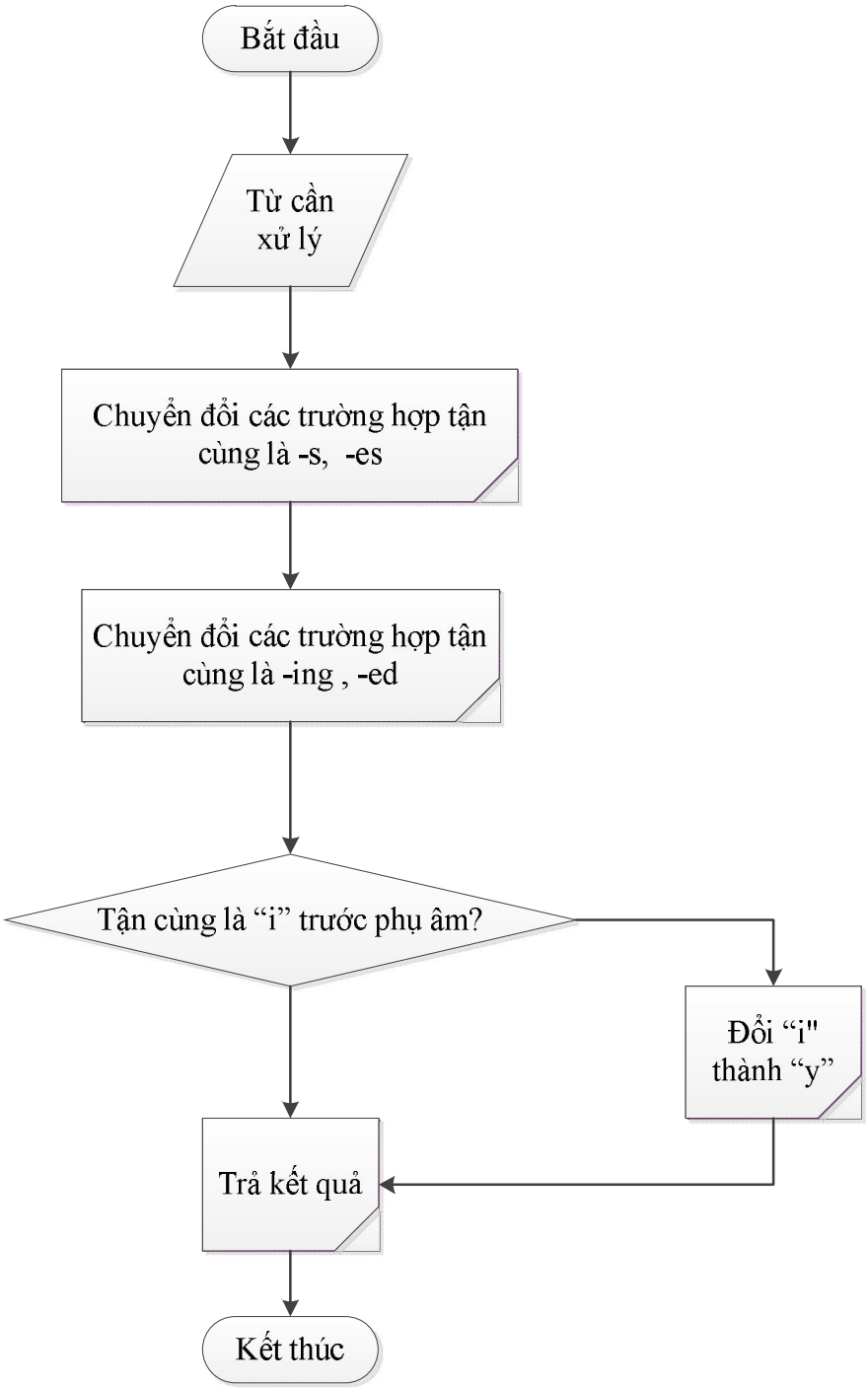
appliance, applicability, applicably, applicant, application,… Trường hợp này sẽ không dùng khôi phục từ gốc. Hậu tố tạo ra biến cách thì sẽ tiến hành đưa về từ gốc. Ví dụ books, booked sẽ đưa về nguyên mẫu là book.

Tóm lại chúng ta chỉ sử dụng khôi phục từ gốc trong trường hợp hậu tố tạo ra biến cách, vì chúng có cùng ngữ nghĩa. Trong trường hợp này ứng dụng sử dụng thuật toán khôi phục từ gốc **Porter [4]** để khôi phục từ gốc.

Thuật toán **Stemming Porter [4]** do Martin Poter đưa ra năm 1980 sau này được tiếp tục phát triển và sử dụng rộng rãi. Thuật toán này có thể giải quyết tất cả các trường hợp để đưa về dạng từ gốc nguyên mẫu. Trong phạm vi ứng dụng này chỉ sử dụng thuật toán cho các trường hợp sau:

- Danh từ ở dạng số nhiều, bỏ -s hoặc –es đưa về nguyên mẫu.
- Động từ chia ở ngôi thứ ba số ít bỏ –s hoặc –es đưa về nguyên mẫu.
- Những từ thêm –ing hoặc –ed được đưa về nguyên mẫu.
- Chuyển "i" thành "y" trong trường hợp gốc từ có nguyên âm. Ví dụ: companies → compani → company.

Sơ đồ hình 4.2 minh họa về thuật toán **Stemming** được sử dụng trong ứng dụng.



Hình 4.2 Sơ đồ thuật toán khôi phục từ gốc

Để thuận tiện tùy theo mục đích của người sử dụng, trong ứng dụng cho phép người dùng tùy chỉnh thiết lập cấu hình: không sử dụng stemming, sử dụng

stemming cho các trường hợp trên (mặc định đã sử dụng), sử dụng stemming khôi phục tận gốc.

4.3. Tìm từ gần đúng

Kết quả nhận diện từ của bộ Tesseract tuy khá cao nhưng vẫn có một số từ nhận diện bị sai do phụ thuộc vào chất lượng ảnh chụp từ văn bản. Lúc này người dùng phải chụp lại phần văn bản hoặc trực tiếp chỉnh sửa kết quả nhận dạng. Chính vì thế việc áp dụng bài toán tìm từ gần đúng vào chương trình nhằm làm tăng tính tiện dụng cho người dùng và làm khắc phục một phần quá trình nhận diện từ không chính xác của bộ Tesseract. Sau đây là các phương pháp có thể áp dụng vào bài toán tìm từ gần đúng trong luận văn.

4.3.1. Khoảng cách Levenstein

Trong khoa học máy tính, khoảng cách **Levenstein [7]** là một đại lượng dùng để đo lường sự khác nhau giữa 2 chuỗi: chuỗi nguồn **s** và chuỗi đích **t**. Khoảng cách Leveinstein giữa 2 chuỗi này được tính bằng số lần biến đổi tuần tự từ chuỗi **s** thành chuỗi **t**. Có 3 phép biến đổi từ chuỗi **s** sang chuỗi **t** là: thêm, xóa và thay thế từng ký tự trong chuỗi **s**.

Ví dụ: khoảng cách Levenstein giữa 2 chuỗi kitten và sitting là 3 vì phải thực hiện tuần tự 3 phép biến đổi từ chuỗi kitten sang sitting:

- Kitten → sitten (thay thế k bằng s)
- Sitten → sittin (thay thế e bằng i)
- Sittin → sitting (thêm g vào cuối chuỗi)

Sau đây là mã giả để minh họa thuật toán tìm khoảng cách Levenstein giữa 2 chuỗi **s** với chiều dài chuỗi là **m** và chuỗi **t** với chiều dài chuỗi là **n**:

```
Int LevensteinDistance ( char s[1...m], char t[1…n])
{
//khởi tạo mảng 2 chiều d và kết quả D[i, j] sẽ là khoảng cách
//Leveinstein giữa 2 chuỗi s và t. Với i, j lần lượt là ký tự đầu tiên
//của chuỗi s và t. Và mảng D sẽ chứa (m+1)(n+1) giá trị.
```

```
D[i,j] :=0 //Khởi tạo các giá trị trong mảng =0
Lặp từ i=1 đến m
D[i,0] :=i
Lặp từ j=1 đến n
D[0,j] :=j
Lặp từ j=1 đến n
        Lặp từ i=1 đến m
        {
        If s[i] = t[j]
        D[i, j] :=D[i-1, j-1]
        Else
        D[i, j] := minimum (
        D[i-1,j] +1 //xóa ký tự
        D[i,j-1] //thêm ký tự
        D[i,j] //thay thế 1 ký tự        )
        }
        Return D[m, n]
}
```

Bảng 4.1 Minh họa ma trận kết quả sau khi tính khoảng cách Levenstein

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	**1**	2	3	4	5	6
i	2	2	**1**	2	3	4	5
t	3	3	2	**1**	2	3	4
t	4	4	3	2	**1**	2	3
i	5	5	4	3	2	**2**	3
n	6	6	5	4	3	3	**2**
g	7	7	6	5	4	4	**3**

Độ phức tạp của thuật toán tìm khoảng cách Leveinstein giữa 2 chuỗi là **O(m*n)** với m, n là độ dài lần lượt của 2 chuỗi. Để áp dụng thuật toán trên vào trong bài toán tìm từ gần đúng ta làm như sau:

- Giả sử ta được kết quả nhận diện từ là chuỗi s với độ dài xác định.
- Ta so sánh luần lượt chuỗi **s** với các từ đã có trong từ điển, lấy các từ có cùng độ dài với nó và đưa vào mảng chuỗi kết quả.
- Sau đó, ta tính khoảng cách Leveinstein của chuỗi **s** với từng từ có trong mảng kết quả. Lấy từ nào có khoảng cách Levenstein là 1 thì cho vào danh sách từ gần đúng.
- Hiển thị các từ có trong danh sách từ gần đúng.

Trên đó là phương pháp sử dụng khoảng cách Leveinstein để tìm từ gần đúng trong từ điển.

Ưu điểm: Phương pháp này đã giải quyết được bài toán tìm từ gần đúng và liệt kê ra các từ có trong danh sách.

Nhược điểm: Tốc độ thực thi chương trình sẽ chậm vì phải dùng vòng lặp để duyệt qua tất các từ có trong từ điển mà số lượng từ thì rất lớn và sau đó mới áp dụng thuật toán để tính khoảng cách Leveinstein.

4.3.2. Thay thế các ký tự gần đúng

Phương pháp áp dụng khoảng cách Leveinstein được dùng nhiều để tính sự khác biệt giữa hai chuỗi ký tự. Tuy nhiên để áp dụng phương pháp này vào trong chương trình để tìm từ gần đúng thì đã mắc phải nhược điểm nêu trên đó là tốc độ thực thi cho việc tìm từ sẽ rất chậm chính vì thế mà trong luận án nhóm chúng em sử dụng thuật toán đơn giản hơn để tìm từ gần đúng trong từ điển dựa vào ý tưởng lần lượt thay thế từng ký tự trong bảng chữ cái Latinh gồm 26 ký tự từ a-z. Mặc định thuật toán chỉ thay thế lần lượt một ký tự từ đầu cho đến cuối chuỗi. Sau đây là mô tả chi tiết giải thuật. Thuật toán nhận đầu vào là một chuỗi và kết quả trả về là danh sách các từ gần đúng chuỗi đầu vào:

- Khởi tạo mảng danh sách các ký tự bao gồm 26 chữ cái thường từ a-z (mặc định thuật toán chỉ làm việc trên các ký tự thường).

- Thực hiện việc lặp từng ký tự trong chuỗi đầu vào. Ứng với từng ký tự trong chuỗi đó sẽ thực hiện thay thế lần lượt 26 ký tự trong mảng ký tự ban đầu.
- Với mỗi kết quả thu được là một từ mới khi thay thế lần lượt từng ký tự, ta sẽ kiểm tra từ mới này có trong dữ liệu từ điển hay không. Nếu có thì ta sẽ đưa từ mới này vào trong mảng danh sách từ.
- Từ mảng danh sách kết quả là các từ gần đúng, ta hiển thị các từ ra cho người dùng chọn.

Ưu điểm: Thực hiện đúng chức năng là tìm từ và liệt kê ra danh sách các từ gần đúng với kết quả nhận dạng có trong dữ liệu. Tốc độ thực thi thuật toán tương đối nhanh và có thể áp dụng để cài đặt trong chương trình.

Nhược điểm: Một số từ chạy không ổn định và phụ thuộc nhiều vào độ dài của từ. Nếu từ ngắn tốc độ tìm từ sẽ rất nhanh và từ dài sẽ mất nhiều thời gian để thực hiện hơn. Tuy nhiên, nhìn vào đánh giá chung cho hai phương pháp trên, phương pháp thay thế ký tự gần đúng mang nhiều ưu điểm hơn và đặc biệt là khả năng ứng dụng thực tế trong chương trình cho bài toán tìm từ gần đúng vì tốc độ thực thi cao hơn rất nhiều.

4.4. Cấu trúc dữ liệu từ điển

Mỗi một từ trong từ điển đều định dạng gồm có từ gốc và nghĩa của từ (bao gồm cả phiên âm và từ loại). Mỗi từ sẽ có từ gốc và nghĩa với kích thước lưu trữ khác nhau. Bảng 4.2 mô tả về các trường dữ liệu này.

Bảng 4.2 Bảng mô tả các trường dữ liệu

STT	Tên trường dữ liệu	Ghi chú
1	Từ gốc	Là từ khóa trong dữ liệu từ điển, độ dài các từ có thể khác nhau nên nó có kích thước biến động.
2	Nội dung	Bao gồm phiên âm, từ loại và các nghĩa của từ. Nó có kích thước biến

		động

Với các trường dữ liệu như vậy, ta phải tổ chức cấu trúc dữ liệu để lưu trữ chúng sao cho dễ dàng truy xuất. Ta có một số giải pháp tổ chức mục từ như sau:

- Tổ chức các mục từ có cùng kích thước cố định.
- Tổ chức các mục từ có kích thước biến động.

Ta sẽ xem xét từng phương pháp cụ thể sau đây.

4.4.1. Tổ chức các mục từ có cùng kích thước cố định.

Để tổ chức các mục từ có kích thước cố định thì chúng ta phải biết kích thước lớn nhất của mục từ có thể có được, để lưu trữ bao quát hết dữ liệu.

Ưu điểm: nhanh chóng, dễ dàng truy xuất dữ liệu của từ khi biết vị trí bắt đầu của nó.

Khuyết điểm: lãng phí bộ nhớ vì có những từ kích thước rất nhỏ nhưng dùng trường dữ liệu lớn gây lãng phí. Điều này rất quang trọng trên thiết bị di động hạn chế về bộ nhớ.

Việc qui định kích thước cố định của mục từ hạn chế việc lưu trữ các mục từ thông dụng có kích thước từ gốc lớn hơn hoặc nội dung lớn hơn. Làm cho việc lưu trữ có vẻ không tự nhiên. Khó để cập nhập thêm mục từ trong quá trình phát sinh dữ liệu từ điển mới.

4.4.2. Tổ chức các mục từ có kích thước biến động.

Tùy theo mỗi mục từ có kích thước bao nhiêu ta sẽ cấp cho nó một bộ nhớ đủ lưu trữ dữ liệu của chúng. Như vậy mỗi mục từ sẽ có thông tin về vị trí bắt đầu và kích thước trường dữ liệu đi kèm theo.

Ưu điểm: tiết kiệm tối đa được tài nguyên bộ nhớ. Không hạn chế kích thước của mục từ. Mục từ có thể có dữ liệu lớn nhỏ tùy ý. Không ảnh hưởng đến các mục từ khác nên có vẻ tự nhiên hơn.

Khuyết điểm: do mục từ có kích thước khác nhau nên cần thêm trường dữ liệu để quản lý kích thước, khó khăn cho việc tra cứu do đó cần tổ chức dữ liệu để hỗ trợ tra cứu nhanh hơn.

Như vậy trên môi trường đi động thì vấn đề bộ nhớ phải được ưu tiên trước do đó tổ chức mục từ có cùng kích thước cố định bộc lộ nhiều khuyết điểm hơn. Trong khi đó tổ chức các mục từ có kích thước biến động sẽ tránh lãng phí không gian bộ nhớ nên giải pháp này được sử dụng trong ứng dụng. Và để giải quyết vần đề tốc độ trong việc truy xuất nó ta phải tổ chức dữ liệu từ điển cho hợp lý.

4.4.3. Tổ chức dữ liệu từ điển tra cứu nhanh

Vấn để tổ chức mục từ dữ liệu động đã được giải quyết. Bây giờ chúng ta giải quyết vần đề tổ chức tập tin để hỗ trợ tìm kiếm. Việc tổ chức cấu trúc tập tin hỗ trợ tìm kiếm nhanh có nhiều phương pháp khác nhau. Sau đây là một số phương pháp tổ chức tập tin:

- Tập tin tuần tự: là tập tin lưu trữ các mục từ liên tiếp nhau. Khi tìm kiếm từ khóa thì thực hiện bằng cách đem từ khóa so sánh tìm kiếm trong các từ gốc của tập tin. Như vậy trường hợp nhanh nhất là từ khóa khớp ngay lần tìm kiếm đầu tiên trong tập tin, và xấu nhất là nó phải duyệt tất cả các từ gốc gần hết tập tin mới tìm ra. Phương pháp này dễ cài đặt tuy nhiên sẽ tốn thời gian xử lý do nguồn dữ liệu rất nhiều từ dẫn đến việc truy xuất chậm hơn đặc biệt trên thiết bị di động.
- Tập tin chỉ mục: để làm tăng hiệu quả tìm kiếm đối với tập tin có kích thước lớn người ta sử dụng tập tin chỉ mục. Tập tin chỉ mục gồm có từ khóa và các thông tin để miêu tả vị trí của dữ liệu trong tập tin ngữ nghĩa. Như vậy việc tìm kiếm từ khóa trở nên dễ dàng hơn khi chỉ cần tìm kiếm trên một tập tin toàn từ khóa và chỉ cần dựa trên thông tin vị trí kèm theo đó ta lấy được ngữ nghĩa. Ta có thể sử dụng tìm kiếm nhị phân để tăng tốc độ tìm kiếm trên tập tin chỉ mục này.
- Tập tin băm: thay vì tìm kiếm trên toàn bộ các từ khóa, ta sử dụng tập tin băm để phân loại các từ khóa có cùng một tính chất nào đó vào cùng một cụm, để giới hạn phạm vi tìm kiếm khi số lượng mẫu tin lớn.
- Cây nhị phân tìm kiếm: ta có thể đọc tất cả các mẫu tin chỉ mục rồi phát sinh ra cây tìm kiếm, rồi lưu cây đó lên tập tin. Việc sử dụng cây nhị

phân tìm kiếm có ưu điểm là khai thác được tốc độ tìm kiếm, tuy nhiên cũng có nhược điểm là phải sử dụng nhiều bộ nhớ lưu trữ cây nhị phân.

Qua việc xem xét các phương pháp tổ chức tập tin trên thì xét theo mặt tốc độ và bộ nhớ, ta nhân thấy việc kết hợp phương pháp chỉ mục và băm tập tin là thích hợp nhất cho việc tìm kiếm nhanh. Sau đây ta sẽ đi vào cụ thể cách sử dụng phương pháp này.

4.4.3.1. Tổ chức tập tin chỉ mục kết hợp băm tập tin

Mục đích việc băm tập tin chỉ mục nhằm chia nhỏ phạm vi tìm kiếm, hỗ trợ tìm kiếm nhanh. Nên nó phải thỏa mãn các tiêu chí sau:

- Việc tính toán hàm băm phải nhanh.
- Các từ khóa phân bố đều trong bảng băm.
- Vẫn giữ thứ tự như dữ liệu từ điển ban đầu.

Tập tin chỉ mục đã được tổ chức sắp xếp tăng dần theo bảng chữ cái nên ta sẽ băm theo các chữ cái đầu của từ. Như vậy hàm băm đã đáp ứng được hai tiêu chí. Để thỏa mãn các tiêu chí còn lại ta có thể băm tập tin chỉ mục theo một hay nhiều cấp tương ứng với số ký tự đầu. Việc chia theo bảng băm càng nhiều cấp sẽ làm phân hóa càng chi tiết dữ liệu. Do đó ta phải xem xét chọn cấp bảng băm cho hợp lý.

- Bảng băm cấp một: Gom các từ có cùng một ký tự đầu tiên thành một cụm. Như vậy dữ liệu sẽ được phân thành khoảng 30 cụm, tuy nhiên sẽ có cụm có nguồn dữ liệu nhiều và cũng có cụm ít dữ liệu. Số lượng dữ liệu trong mỗi cụm vẫn còn lớn gây ra cảm giác ứng dụng chạy chậm nên cần phải phân hoạch nhỏ hơn nữa.
- Bảng băm cấp hai: Các từ có cùng hai ký tự đầu sẽ được gom thành một cụm, nên phạm vi tìm kiếm đã giảm đáng kể. Các phân hoạch sẽ nhỏ hơn nên thời gian tìm kiếm trong khoảng phân hoạch đó giảm đáng kể.
- Bảng băm cấp ba: Các từ có cùng ba ký tự đầu sẽ được gom thành một cụm. Các tập tin sẽ được chia vụn, kích thước bộ nhớ và thời gian nạp từ sẽ tăng lên.

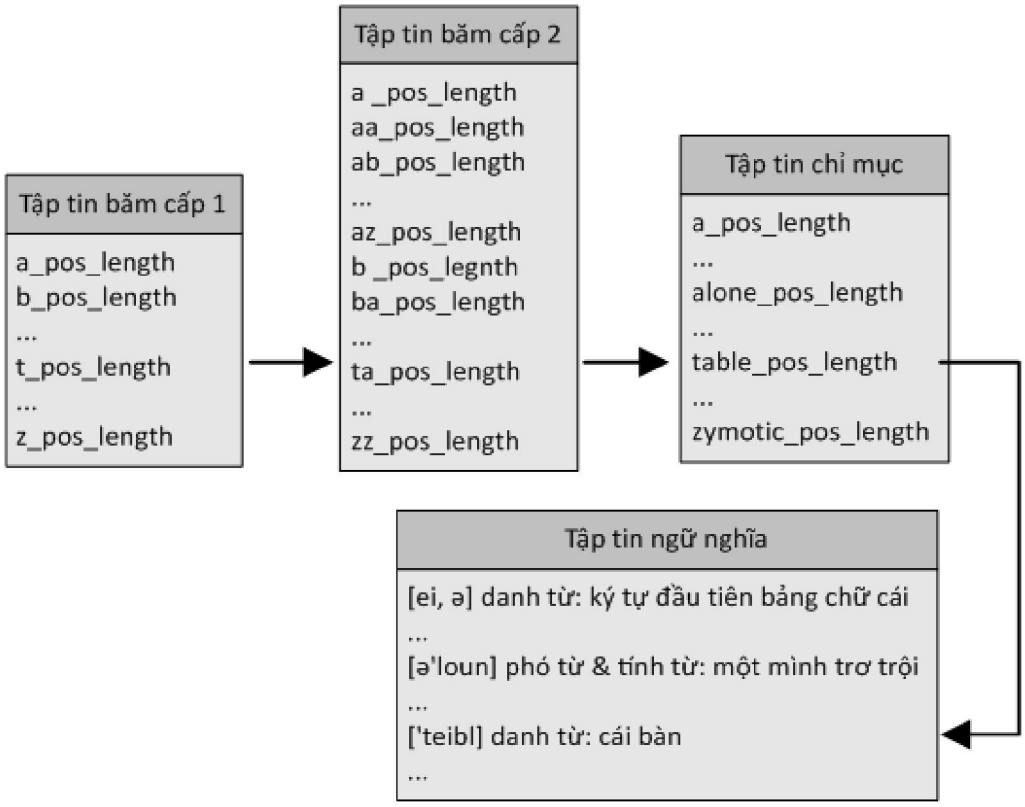
Như vậy việc băm cấp một sẽ gây ra cảm giác chậm, còn nếu băm cấp ba hoặc nhiều hơn thì có thể chia quá vụn tập tin và tăng kích thước bộ nhớ một cách không cần thiết. Trong các thiết bị di động hiện nay tốc độ xử lý cũng tăng đáng kể nên việc sử dụng băm cấp hai có thể trung hòa giữa thời gian nạp và kích thước bộ nhớ bị chiếm dụng.

4.4.3.2. Tổ chức tập tin ngữ nghĩa

Dựa vào chỉ mục ta có thông tin về điểm bắt đầu và độ dài của phần nghĩa trong tập tin ngữ nghĩa. Do đó, tập tin ngữ nghĩa chỉ bao gồm các trường dữ liệu sắp xếp tuần tự trong một tập tin. Mỗi trường dữ liệu bao gồm phần phiên âm quốc tế, từ loại và các nghĩa khác của từ.

4.4.3.3. Tra cứu từ điển

Qua việc phân tích cấu trúc tổ chức dữ liệu từ điển như trên ta có sơ đồ dữ liệu từ điển (hình 4.2) và dựa trên đó ta có thuật toán tra từ thích hợp trên cách tổ chức dữ liệu này.



Hình 4.3 Sơ đồ tổ chức tập tin từ điển

- Thuật toán tra từ trên sơ đồ trên như sau. Giả sử ta muốn tra từ "**table**".

  ➢ Bước 1: Đọc **tập tin băm cấp 1** rồi dùng tìm kiếm nhị phân tra từ khóa "**t**" trong tập tin băm cấp 1. Dựa vào từ khóa "t" ta tìm được thông tin vị trí bắt đầu (pos) và độ dài (length) của cụm dữ liệu ở tập tin băm cấp 2.

  ➢ Bước 2: Đọc **tập tin băm cấp 2** tại vị trí pos và chiều dài length vừa tìm được ta lấy ra được là một khối dữ liệu gồm các từ khóa bắt đầu 2 ký tự và có "t" đứng đầu: "ta" → "tw". Tiếp tục tìm kiếm nhị phân từ khóa "**ta**" của "table" trong khối dữ liệu trên ta tìm được thông tin vị trí bắt đầu (pos) và độ dài (length) của cụm dữ liệu ở tập tin chỉ mục.

  ➢ Bước 3:Đọc **tập tin chỉ mục** tại vị trí pos và chiều dài length vừa tìm được ta lấy ra được là một khối dữ liệu gồm các tất cả từ khóa bắt đầu 2 ký tự "ta". Tiếp tục tìm kiếm nhị phân từ khóa "**table**" trong khối dữ liệu

trên ta tìm được thông tin vị trí bắt đầu (pos) và độ dài (length) của cụm dữ liệu ở tập tin ngữ nghĩa.

- Bước 4: Đọc **tập tin ngữ nghĩa** tại vị trí pos và chiều dài length vừa tìm được ta lấy ra được là một khối dữ liệu là phần ngữ nghĩa của từ "table". Trả kết quả về cho ứng dụng.

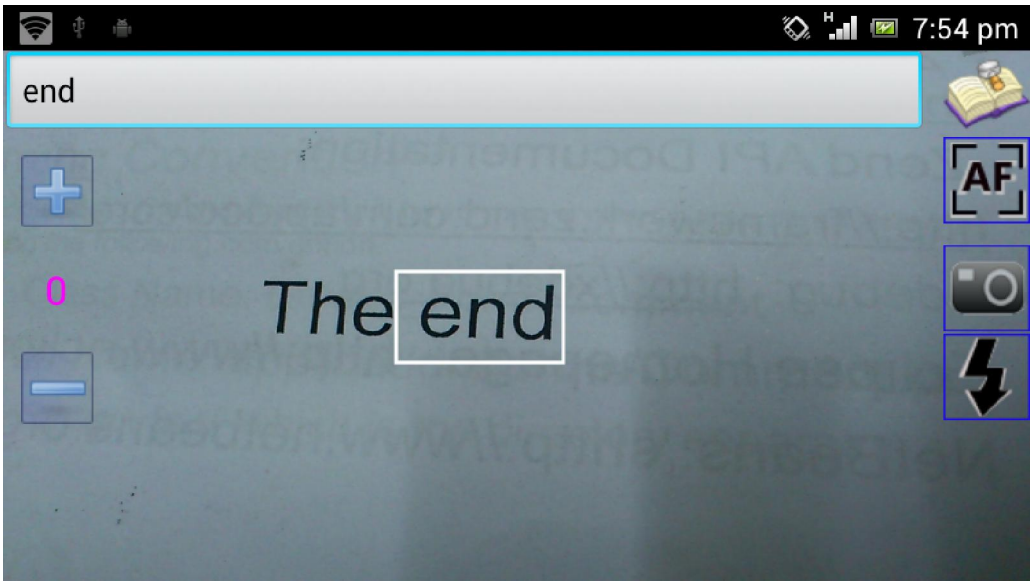
Như vậy với bảng băm hai cấp chúng ta có thể nhanh chóng tìm ta chỉ mục nhờ vào thuật toán tìm kiếm nhị phân trên các phân hoạch con. Sau đó dựa vào chỉ mục có thể tìm được nội dung ngữ nghĩa.

Chương 5 : CÀI ĐẶT VÀ THỰC NGHIỆM ỨNG DỤNG

Khi phát triển ứng dụng từ điển dùng camera trên điện thoại di động Android ngoài các vấn đề về giải thuật, cấu trúc dữ liệu còn có các đặc trưng riêng về màn hình hiển thị, xử lý camera, lưu trữ cấu hình, âm thanh… Do đó trong lập trình ứng dụng cần phải giải quyết các vấn đề kỹ thuật liên quan.

5.1. Vẽ khung và các control trên màn hình camera.

Màn hình camera là màn hình chính của ứng dụng, nó được thiết kế khá đặc biệt gồm có 2 lớp. Lớp thứ nhất bên dưới là hiển thị hình ảnh camera. Lớp thứ hai ở bên trên là các control gồm khung giới hạn chụp ảnh, các button, input text, text view. Có hai vấn đề xảy ra cần giải quyết với hai lớp này: hiển thị và nhận sự kiện tương tác. Hình ảnh 5.1 là giao diện màn hình chính camera.



Hình 5.1 Giao diện màn hình camera

Đối với vấn đề hiển thị cho hai lớp nằm chồng lên nhau, thì trong Android cho phép chúng ta thiết kế layout như vậy nhờ vào **FrameLayout**. Như vậy các control đươc chia thành 2 nhóm để hiển thị trên 2 lớp này. Nhóm 1 là: SurfaceView dùng hiển thị camera. Nhóm 2 gồm có khung giới hạn vùng chụp, các button chụp ảnh,

button làm rõ nét (auto focus), button tắt mở đèn fash, hai button zoom camera, button tra từ và edittext hiển thị từ được chụp. Trên một số dòng thiết bị điện thoại không có hỗ trợ auto focus, hoặc không hỗ trợ đèn flash thì các button tương ứng sẽ không hiển thị.

Đối với vấn đề nhận sự kiện tương tác, do ứng dụng chúng ta cần vẽ một khung hình giới hạn vùng chụp. Khung này có kích thước thay đổi tùy ý của người dùng. Vì thế cần phải có một lớp đồ họa (lớp Paint) bên trên để hỗ trợ việc vẽ khung. Lớp **Paint** vẽ khung sẽ che hết toàn bộ màn hình nên chúng ta sẽ không tương tác trực tiếp được các control bên dưới. Khi click vào các control sẽ không nhận được sự kiện. Để giải quyết vấn đề này, chúng em sử dụng giả lập sự kiên click button, có nghĩa là bắt sự kiện theo tọa độ, vị trí tương tác trên màn hình, nếu vị trí đó nằm trên control nào thì control đó gọi sự kiện click. Nếu nhấp ngay khung và kéo thả thì khung sẽ thay đổi vị trí. Việc xử lý này tuy gây khó khăn nhưng nó bảo đảm tất cả các control đều được tương tác dù nó nằm xếp lên nhau.

Khung chữ nhật trên màn hình chính dùng để giới hạn vùng chụp có thể thay đổi kích thước cho phù hợp với kích thước chữ thực tế để đảm bảo độ chính xác. Trong ứng dụng lớp **RectView** được tạo ra dùng để quản lý công việc này. Để vẽ hình chữ nhật trên màn hình cần phải có một lớp đồ họa đó là lớp **Paint**, ta khởi tạo và khai báo các đối tượng cần thiết như sau:

```
private Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
private static float top;
private static float left;
private static float right;
private static float bottom;
```

Với Paint là lớp chứa các thông tin về kiểu dáng và màu sắc, cung cấp các phương thức dùng để vẽ các hình học, vẽ chữ và các bitmap. Còn các tham số top, left, right, bottom dùng để xác định vị trí trên, trái, phải, dưới của hình chữ nhật.

Hàm dùng để khởi tạo các tham số như sau:

```
private void Init() {
      // TODO Auto-generated method stub
```

```
            left = (MAX_WIDTH / 2) - 100;
            top = (MAX_HEIGHT / 2) - 40;
            right = left + 120;
            bottom = top + 60;
            paint.setColor(Color.WHITE);
            paint.setStrokeWidth(3);
            paint.setStyle(Style.STROKE);
            invalidate();
      }
```

Sau đó chúng ta sẽ gọi phương thức ***onDraw()*** để vẽ hình chữ nhật với các tọa độ trên.

```
    @Override
     protected void onDraw(Canvas canvas) {
            // TODO Auto-generated method stub
            canvas.drawRect(left, top, right, bottom, paint);

     }
```

5.2. Thu nhận ảnh từ camera điện thoại.

Đầu vào cho bộ nhận diện ký tự quang học Tesseract là tập tin hình ảnh dạng bitmap chính vì thế ta cần lập trình xử lý camera trên điện thoại Android để có thể thu nhận ảnh từ văn bản giấy.

Hầu hết các loại điện thoại thông minh hiện nay đều được tích hợp phần cứng camera trong thiết bị. Và camera trở thành phần không thể thiếu trong các hệ điều hành cho di động. Android không phải là một ngoại lệ, và trong thư viện các hàm API được hỗ trợ sẵn trong Android SDK thì Android đã cung cấp cho ta lớp tiện ích để có thể truy xuất và điều khiển camera trên thiết bị có hỗ trợ. Để có thể thực hiện được điều này, ta sử dụng lớp **Camera** cùng với lớp **SurfaceView**().

Lớp **Camera** cung cấp các phương thức để có thể thay đổi các thông số thiết lập trên camera, xem trước ảnh và đặc biệt là ghi nhận hình ảnh từ ống kính camera của điện thoại. Trước khi có thể sử dụng được camera trên thiết bị ta cần phải thiết

lập quyền để sử dụng các phần cứng trong thiết bị và trường hợp này là camera, các quyền được thiết lập sẽ được đặt trong tập tin **AndroidManifest.xml**:

```
<uses-permission android:name="android.permission.CAMERA" />
```

Sau đó ta tạo ra lớp SurfaceView để hiển thị hình ảnh trực tiếp qua ống kính camera điện thoại. Lớp này nắm giữ phần hiển thị hình ảnh và chịu trách nhiệm vẽ lại trên diện tích của màn hình. Ta sử dụng lớp interface **SurfaceHolder** để truy cập phần bề mặt bên dưới lớp SurfaceView. Ta thực thi lại các hàm trong interface SurfaceHolder.Callback và thêm các hàm callback này trong SurfaceHolder. Các phương thức được thực thi lại trong interface SurfaceHolder.Callback bao gồm 3 phương thức sau:

```
public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2,
int arg3) ;
public void surfaceCreated(SurfaceHolder arg0)
public void surfaceDestroyed(SurfaceHolder arg0)
```

Hàm surfaceCreated được gọi sau khi mà một surface đã được tạo ra, hàm surfaceChanged được gọi sau khi có bất kỳ sự thay đổi nào xảy ra trên surface và hàm surfaceDestroyed được gọi khi surface bị hủy.

Sau khi đã tạo ra 1 surface để thể hiện hình ảnh trên màn hình, ta bắt đầu sử dụng lớp **Camera**, gọi phương thức **Camera.open()** để mở ống kính máy ảnh trên điện thoại, sau đó ta thiết lập xem ảnh trực tiếp trên bề mặt thông qua hàm **setPreviewDisplay().** Các hàm này được gọi lần đầu trong hàm callback **surfaceCreated()** khi khởi tạo các thông số ban đầu cho camera.

```
    public void surfaceCreated(SurfaceHolder arg0) {
        Camera mCamera = Camera.open();
        mCamera.setPreviewDisplay(mySurface_holder);
}
```

Sau đó ta sẽ thiết lập các thông số tùy chỉnh trong camera thông qua đối tượng Camera.parameters của lớp **Camera**. Gọi hàm **camera.getParameters**() để lấy các thông số được thiết lập hiện tại trong máy ảnh điện thoại. Ta có thể thiết lập lại các thông số trong camera bằng gọi các phương thức có dạng set…() trong đối

tượng Parameters và hoàn tất việc thay đổi giá trị các thông số dùng phương thức setParameters. Ta cần thiết lập lại các thông số của camera khi hàm callback ***surfaceChanged()*** được gọi. Sau khi thiết lập lại thông số cho camera, gọi phương thức ***startPreview()*** để bắt đầu chế độ xem trước hình ảnh trực tiếp qua ống kính của camera điện thoại.

```
	public void surfaceChanged(SurfaceHolder arg0, int arg1, int
arg2, int arg3) {
	Camera.Parameters params = mCamera.getParameters();
		. . . . .
		// Thực hiện việc thay đổi các thông số ở đây
		. . . . .
		// hoàn tất việc thay đổi các thông số
		mCamera.setParameters(params);
		mCamera.startPreview();
}
```

Sau khi kết thúc quá trình sử dụng camera ta gọi phương thức camera.realease() và dừng phát khung ảnh xem trước stopPreview(). Các phương thức này được sử dụng khi ***surfaceDestroyed()*** được gọi:

```
	public void surfaceDestroyed(SurfaceHolder arg0) {
		// TODO Auto-generated method stub
		mCamera.release();
		mCamera.stopPreview();
	    }
```

Trong lớp **Camera** có hỗ trợ các lớp interface callback đến nhiều sự kiện khác nhau trong ứng dụng. Sau đây là các interface hỗ trợ việc gửi thông báo đến các sự kiện trong lớp **Camera**:

- *Camera.AutoFocusCallback*
- *Camera.ErrorCallback*
- *Camera.PictureCallback*
- *Camera.PreviewCallback*

- *Camera.ShutterCallback*

Interface *Camera*.AutoFocusCallback được sử dụng để gửi thông báo khi quá trình lấy nét tự động (auto focus) hoàn tất. Tính năng lấy nét tự động thường được sử dụng trong camera để tăng chất lượng ảnh và khiến ảnh chụp rõ vật thể hơn. Tuy nhiên không phải thiết bị nào có camera cũng hỗ trợ tính năng này. Trong interface này, hàm ***onAutoFocus()*** là hàm thuần ảo và ta cần thực thi hàm này trong chương trình. Hàm này sẽ được gọi khi quá trình lấy nét tự động hoàn tất. Và để bắt đầu cho camera thực hiện việc lấy nét tự động ta sử dụng phương thức ***camera.autoFocus().***

Interface Camera.ErrorCallback để báo hiệu khi có lỗi xảy ra. Phương thức chính là ***onError()*** sẽ được gọi khi có xảy ra lỗi trong việc thao tác với camera.

Interface **Camera.PictureCallback** được sử dụng để cung cấp dữ liệu ảnh sau khi ảnh đã được chụp. Phương thức chính ***onPictureTaken()*** được gọi khi dữ liệu đã sẵn sàng. Định dạng của ảnh phụ thuộc vào định dạng ảnh của camera và có thể được thiết lập thông qua đối tượng Camera.Parameters.

Interface Camera.PreviewCallback được sử dụng khi cung cấp dữ liệu của khung hình xem trước. Phương thức chính là ***onPreviewFrame()*** được sử dụng khi khung duyệt trước ảnh đã có dữ liệu. Định dạng của dữ liệu cũng phụ thuộc vào định dạng hiện tại của camera.

Cuối cùng là interface Camera.ShutterCallback để thông báo khi ảnh đã được chụp xong từ camera điện thoại. Phương thức chính là ***onShutter()***.

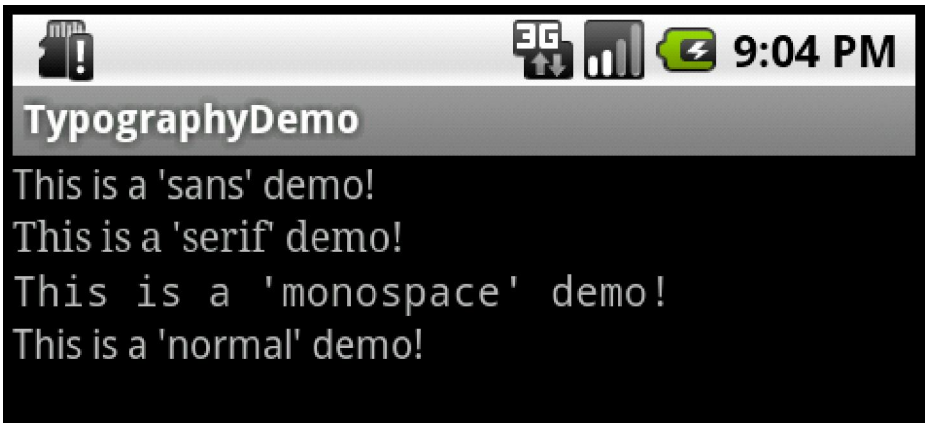
5.3. Hiển thị tiếng Việt và định dạng chữ trên màn hình.

Nội dung ngữ nghĩa của từ điển bao gồm từ khóa, từ phiên âm quốc tế, từ loại, các nghĩa tiếng Việt của từ, các ví dụ sử dụng, các từ đồng nghĩa hoặc các từ liên kết, do đó nội dung phải làm sao cho người dùng dễ nhìn và dễ hiểu nhất, với nhiều phần trong ngữ nghĩa như vậy chúng ta phải định dạng kiểu chữ, màu chữ, cỡ chữ cho phù hợp.

5.3.1. Hiển thị tiếng Việt trên Android

Với sự ra đời của font chữ Unicode (một ký tự biểu diễn bằng 2 byte) có các ký tự tiếng Việt trong phần mở rộng, tiếng Việt đã được hiển thị tốt như các ngôn ngữ khác trong những phần mềm máy tính. Đối với các thiết bị kỹ thuật số cá nhân khác như martphone, di động… cũng sử dụng cùng loại font. Do đó khi phát triển từ điển trên các thiết bị này thì vấn đề là chỉ cần sử dụng font Unicode trên các control hiển thị.

Trong hệ điều hành Android hiện nay thì các font Unicode đã được sử dụng làm font chữ hiển thị. Trong mỗi thiết bị Android đã hỗ trợ gia đình font Droid chuẩn sau Droid Sans, Droid Sans Mono và Droid Serif. Mặc định bình thường khi không tùy chỉnh loại font nào thì hiển thị là font Droid Sans. (hình minh họa).



Hình 5.2 Minh họa gia đình font Droid

Với font Droid có thể hiển thị được định dạng Unicode. Do đó để hiển thị được đầy đủ các dấu tiếng Việt trong hệ điều hành Android thì nội dung đó cần phải định dạng theo chuẩn Unicode cụ thể là ứng dụng sử dụng mã UTF-8.

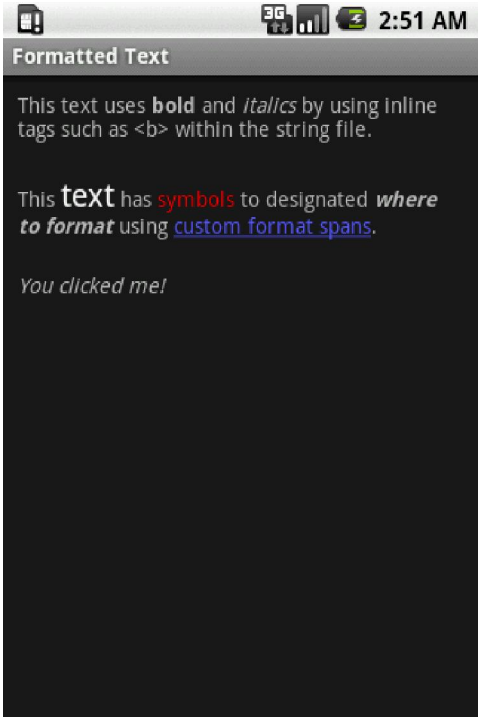
UTF-8 là một cách mã hóa để có tác dụng giống như UTF-16 và UTF-32, UTF-8 có thể biểu diễn tất cả các chữ cái trong bộ ký tự Unicode, nhưng điểm khác biệt quan trọng nhất là UTF-8 được thiết kế để tương thích với chuẩn ASCII. UTF-8 có thể sử dụng từ một (cho những ký tự trong ASCII) cho đến 6 byte để biểu diễn một ký tự.

Chính vì tương thích với ASCII, UTF-8 cực kỳ có lợi thế khi được sử dụng để bổ sung hỗ trợ Unicode cho các phần mềm. Thêm vào đó, các nhà phát triển phần mềm vẫn có thể sử dụng các hàm thư viện có sẵn của ngôn ngữ lập trình C để so sánh (comparisons) và xếp thứ tự. Ngược lại, để hỗ trợ các cách mã hóa 16 bit hay 32 bit như ở trên, một số lớn phần mềm buộc phải viết lại do đó tốn rất nhiều công sức. Một điểm mạnh nữa của UTF-8 là với các văn bản chỉ có một số ít các ký tự ngoài ASCII, hay thậm chí cho các ngôn ngữ dùng bảng chữ cái Latinh như tiếng Việt, tiếng Pháp, tiếng Tây Ban Nha, v.v… cách mã hóa kiểu này cực kỳ tiết kiệm không gian lưu trữ. UTF-8 được thiết kế đảm bảo không có chuỗi byte của ký tự nào lại nằm trong một chuỗi của ký tự khác dài hơn. Điều này khiến cho việc tìm kiếm ký tự theo byte trong một văn bản là rất dễ dàng.

Bây giờ việc hiển thị các font chữ lên các control ta chì cần gọi hàm ***SetText(myText)*** và tiếng Việt sẽ hiển thị bình thường.

5.3.2. Định dạng ngữ nghĩa từ điển.

Như đã giới thiệu nhiều phần trong ngữ nghĩa của từ điển bao gồm từ khóa, từ phiên âm quốc tế, từ loại, các nghĩa tiếng Việt của từ, các ví dụ sử dụng, các từ đồng nghĩa như vậy chúng ta phải định dạng kiểu chữ, màu chữ, cỡ chữ cho dễ phân biệt các thành phần trên.



Hình 5.3 Hình Định dạng văn bản hiển thị theo các kiểu phong cách

Trong Android các TextView không dễ dàng thay đổi phong cách định dạng của một chuỗi văn bản, giống như không có hỗ trợ sẵn để thực hiện chức năng sau: `textView.setTextColor(Color.RED, 10, 20);` nhằm thiết lập văn bản từ ký tự thứ 10 đến ký tự thứ 20 có màu đỏ, mà phải có các phương pháp khác định dạng chữ hiển thị không chỉ với màu sắc mà còn với tất cả các kiểu dáng khác.

➢ Phương pháp thứ nhất: sử dụng thẻ HTML định dạng chuỗi. Khi đó nội dung từ điển phải được định dạng sẵn các thuộc tính của thẻ, việc làm này tương tự như định dạng thẻ trong web. Chúng ta có thể sử dụng các thẻ HTML <b>, <u>, <i>,… để định dạng các kiểu chữ đậm, gạch chân, chữ nghiêng,… Tuy nhiên phương pháp này sẽ gây khó khăn cho nguồn dữ liệu lớn mà chưa được thiết kế sẵn các thẻ tag. Ví dụ như sau: khi các bạn có các nội dung được định dạng như sau:

```
<string name="text1">This text uses <b>bold</b> and <i>italics</i> by using inline tags such as <b> within the string file.</string>
```

Thì nội dung này sẽ được hiển thị như sau:

This text uses **bold** and *italics* by using inline tags such as <b> within the string file.

➢ Phương pháp thứ hai dùng CharSequence. Trong Android, Textview dùng để định dạng và hiển thị văn bản không chỉ với kiểu dữ liệu lớp String đơn giản mà còn có thể sử dụng lớp CharSequence. Một CharSequence là một lớp đối tượng, nó trừu tượng hơn String, String là một sub-class của CharSequence. Một CharSequence là một dãy các ký tự giống như String nhưng nó có thể định dạng một loạt các ký tự bên trong, như là một SpannableString. Những gì chúng ta cần làm là thay đổi dãy ký tự ở giữa TextView để thêm vào một khoảng định dạng trong chuỗi đó. Nói một cách chính xác hơn là chúng ta sẽ thêm CharacterStyles vào CharSequence của TextView, cái mà được gọi là SpannableString.

So sánh hai phương pháp trên ta thấy phương pháp thứ hai đơn giản hơn vì không cần phải định dạng thẻ HTML phức tạp, và do trong dữ liệu đã có các ký hiệu đánh dấu phân biệt các thành phần ngữ nghĩa, nên chỉ cần xác định vị trí chuỗi cần định dạng là có thể định dạng chính xác theo ý muốn. Do đó trong ứng dụng này, em đã chọn phương pháp dùng CharSequences để định dạng.

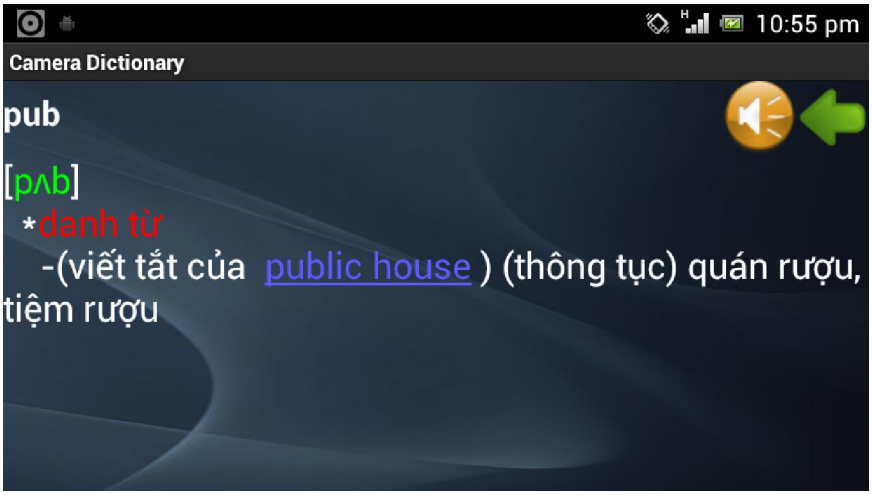
Đây là một phương pháp định dạng văn bản một cách động vì chỉ cần ta xác định vị trí bắt đầu và kết thúc trong cần định dạng trong chuỗi. Việc xác định dựa vào ký hiệu khóa đánh dấu. Ví dụ một TextView có nội dung như sau: “Hello #World#!”. Bây giờ chúng ta muốn chữ “World có màu đỏ thì phải tìm vị trí của hai dấu “#” sau đó dựa vào Class **SpannableStringBuilder** để gọi hàm ***setSpan()*** định dạng đoạn text đó.

```
     public CharSequence FormatText() {
          CharSequence myText = "Hello #World#!";
          int start = myText.toString().indexOf("#");
          int end = myText.toString().indexOf("#");
          // Copy the spannable string to a mutable spannable
string
          SpannableStringBuilder ssb = new
SpannableStringBuilder(myText);
```

```
            ssb.setSpan(new  ForegroundColorSpan(Color.RED),  start,
end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
            // Delete the tokens before and after the span
            ssb.delete(start, start + 1);
            ssb.delete(end, end + 1);
            myText = ssb;
            return myText;
}
```

Trong ví dụ trên ForegroundColorSpan(color) kiểu định dạng tô màu nền. Ngoài ra còn có các kiểu in đậm (TypeFace.BOLD), in nghiêng (TypeFace.ITALIC), hoặc vừa đậm vừa nghiêng (TypeFace.BOLD_ITALIC), kiểu gạch chân (UnderlineSpan), và kiểu đường dẫn liên kết (ClickableSpan).

Kiểu đường dẫn liên kết (ClickableSpan) dùng để liên kết các từ có liên quan với nhau (từ đồng nghĩa, từ trái nghĩa, từ viết tắt …) trong phần nội dung ngữ nghĩa. Khi ta click vào từ thì nó sẽ tiếp tục hiển thị nghĩa. Ví dụ: "pub" là viết tắt của "public house". Khi tra từ "pub" sẽ có đường dẫn liên kết đến nghĩa từ "public house".



Hình 5.4 Hình định dạng liên kết

Muốn sử dụng ClickSpan ta phải khai báo đăng ký sử dụng.

```
MovementMethod m = myText.getMovementMethod();
if ((m == null) || !(m instanceof LinkMovementMethod)) {
```

```
myText.setMovementMethod(LinkMovementMethod.getInstance());
}
```

Sau đó ta chọn đoạn text cần link kết và bắt sự kiện click vào text:

```
     final String textClicked = myText.substring(start, end);
     ssb.setSpan(new ClickableSpan() {
          public void onClick(View widget) {
// TODO Auto-generated method stub
     MeanWordActivity.restartActivity(textClicked, widget);
                    }
          }
          , start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

5.4. Mã hóa dữ liệu từ điển.

Hiện nay vấn đề bảo mật, mã hóa thông tin ngày càng quan trọng và rất cần thiết, nó có nhiều ứng dụng rộng rãi trong cuộc sống, phổ biến trong các lĩnh vực khác nhau. Dữ liệu trong ứng dụng từ điển cũng rất quan trọng, nó phải được mã hóa để tránh người dùng sao chép dữ liệu vi phạm bản quyền dữ liệu. Đặc biệt, khi những kẻ xấu lợi dụng dữ liệu thay đổi nội dung dữ liệu bên trong từ điển sẽ gây ra những hậu quả nghiêm trọng, làm lệch lạc thông tin cho người sử dụng. Do đó, dữ liệu trong từ điển phải được mã hóa để tăng độ an toàn, bảo mật thông tin.

Trong thực tế phương pháp mã hóa **DES [5] (Data Encryption Standard)** được sử dụng rộng rải nhất. Nó thường được dùng để mã hóa dòng dữ liệu trên mạng và dữ liệu lưu trữ trên đĩa. DES được FIPS (Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976. Sau đó chuẩn này được sử dụng rộng rãi trên phạm vi thế giới.

DES là thuật toán mã hóa khối: nó xử lý từng khối thông tin của bản rõ có độ dài xác định và biến đổi theo những quá trình phức tạp để trở thành khối thông tin của bản mã có độ dài không thay đổi. Trong trường hợp của DES, độ dài mỗi khối là 64 bit. DES cũng sử dụng khóa để cá biệt hóa quá trình chuyển đổi. Nhờ vậy, chỉ khi biết khóa mới có thể giải mã được văn bản mã. Khóa dùng trong DES có độ dài

toàn bộ là 64 bit. Tuy nhiên chỉ có 56 bit thực sự được sử dụng; 8 bit còn lại chỉ dùng cho việc kiểm tra. Vì thế, độ dài thực tế của khóa chỉ là 56 bit. Ở thời điểm DES ra đời người ta đã tính toán rằng việc phá được khoá mã DES là rất khó khăn, nó đòi hỏi chi phí hàng chục triệu USD và tiêu tốn khoảng thời gian rất nhiều năm. Cùng với sự phát triển của các loại máy tính và mạng máy tính có tốc độ tính toán rất cao, khoá mã DES có thể bị phá trong khoảng thời gian ngày càng ngắn với chi phí ngày càng thấp. Dù vậy việc này vẫn vượt xa khả năng của các hacker thông thường và mã hoá DES vẫn tiếp tục tồn tại trong nhiều lĩnh vực như ngân hàng, thương mại, thông tin...

Trong ứng dụng dữ liệu đã được sử dụng thuật toán mã hóa DES, để mã hóa tập tin ngữ nghĩa. Mỗi một từ khóa trong tập tin chỉ mục xác đinh một dãy byte nội dung nghĩa của từ khóa đó trong tập tin ngữ nghĩa. Tuy nhiên nội dung đó đã được mã hóa theo thuật toán DES, với key nằm trong 8 byte cuối của dãy nội dung. Do đó ứng dụng muốn hiển thị đúng nội dung ngữ nghĩa phải tách key đó ra và dùng key đó giải mã ngược các nội dung còn lại, ta được nội dung thật sự.

Trong Java có các thư viện **java.security** và **javax.crypto** hỗ trợ tạo key và cài đặt sẵn các giải thuật mã hóa dữ liệu theo chuẩn. Ứng dụng sử dụng các thư viện này để mã hóa theo giải thuật DES. Quá trình tạo key (dùng đối tượng KeyGenerator) và mã hóa dữ liệu (dùng đối tượng Cipher) như sau, kết quả trả về là dữ liệu đã được mã hóa và key kèm theo [6]:

```
public byte []EnCrypt(byte []data){
    key = null;
    try {
        KeyGenerator kg = KeyGenerator.getInstance("DES");
        Cipher cipher = Cipher.getInstance("DES");
        key = kg.generateKey();
        cipher.init(Cipher.ENCRYPT_MODE, key);
        return (cipher.doFinal(data));
    } catch (Exception e) {
        return null;
```

```
        }
    }
```

Quá trình giải mã dữ liệu dựa vào key như sau, kết quả trả về sẽ là dữ liệu được giải mã:

```
public byte []DeCrypt(byte []data, Key key){
    try {
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE, key);
        return cipher.doFinal(data);
    } catch (Exception e) {
        return null;
    }
}
```

Như vậy với việc mã hóa dữ liệu với thuật toán DES giúp đảm bảo dữ liệu từ điển trở nên an toàn hơn. Tránh bị đánh cắp và thay đổi dữ liệu.

5.5. Lưu trữ cấu hình chức năng của ứng dụng.

Các thông số kỹ thuật của ứng dụng được lưu trữ cục bộ trong thiết bị di động. Có nhiều cách lưu trữ thông tin trên di động tùy thuộc vào yêu cầu sử dụng, có thể lưu thông tin trên tập tin kiểu text hoặc xml rồi ứng dụng đọc tập tin đó, cách này gồm có lưu trữ trên bộ nhớ thiết bị di động và lưu trữ trên bộ nhớ các thiết bị lưu trữ bên ngoài (ví dụ như thẻ nhớ), hoặc có thể lưu trữ trên hệ cơ sở dữ liệu SQLite hiện đã được hỗ trợ cho Android, khi cần đọc dữ liệu thì thực hiện câu truy vấn cơ sở dữ liệu để lấy thông tin, hoặc có thể lưu trữ trên web với máy chủ (server) là của chính mình. Các cách trên chúng ta đều cần lưu trữ trên tập tin và phải tự thiết kế các đối tượng và các control giao diện tương ứng. Thực ra trong Android đã hỗ trợ sẵn cho chúng ta một đối tượng giúp cho chúng ta có thể dễ dàng lưu trữ các thông tin cấu hình của ứng dụng chỉ cần chúng ta thao tác trực tiếp trên các control đại diện đó chính là **SharedPreferences**.

Lớp SharedPreferences là khuôn khổ (framework) chung cho phép chúng ta ghi và đọc cặp đôi dữ liệu "từ khóa – giá trị" (key-value). Với SharedPreferences

chúng ta có thể lưu trữ nhiều kiểu dữ liệu như: boolean, int, long, float và string. Các kiểu dữ liệu này sẽ kéo dài xuyên suốt các phiên của người sử dụng thậm chí ngay cả khi ứng dụng đã bị tắt.

SharedPreferences hỗ trợ các loại control giao diện (**PreferenceScreen**) cho người dùng dễ dàng thực hiện các chức năng tùy chỉnh lưu trữ thông tin tương ứng với các kiểu dữ liệu. Có nhiều loại control trong **PreferenceScreen** tương tự các control cơ bản trong View của Android như: EditTextPreference dùng để lưu trữ các chuỗi text, CheckBoxPreference dùng để lưu trữ giá trị logic (boolean), ListPreference dùng để lưu trữ giá trị đã chọn trong danh sách lựa chọn, v.v… Mỗi đối tượng control giao diện trên đều có một tên key tương ứng để lưu trữ giá trị. Khi chúng ta thao tác trên các control này thì dữ liệu sẽ tự động lưu trữ xuống thiết bị dựa theo key.

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:defaultValue="true"
        android:key="spelling"
        android:summary="Chức năng tìm từ gần đúng"
        android:title="Spelling suggestion" />
    <ListPreference
        android:defaultValue="2"
        android:entries="@array/stemmer_display"
        android:entryValues="@array/stemmer_value"
        android:key="stemmer"
        android:summary="Chức năng khôi phục từ gốc"
        android:title="Stemming" />
    <ListPreference
        android:defaultValue="18"
        android:entries="@array/font_size_display"
        android:entryValues="@array/font_size_value"
        android:key="fontsize"
```

```
        android:summary="Chỉnh cỡ chữ của hiển thị từ điển"
        android:title="Font size of dictionary" />
</PreferenceScreen>
```

Hình 5.5 ScreenPreference **Hình 5.6 ListPreference**

Khi ứng dụng muốn đọc các giá trị thì trước hết ứng dụng phải đăng ký sử dụng SharedPreferences.

```
SharedPreferences prefs;
prefs=PreferenceManager.getDefaultSharedPreferences(
getApplicationContext());
```

Ví dụ bây giờ ứng dụng muốn đọc giá trị của CheckBoxPreference thì gọi getBoolean, và tương tự cho các kiểu dữ liệu khác.

```
if (prefs.getBoolean("spelling", true)) {
//do something
}
```

Trong ứng dụng từ điển camera này em đã sử dụng SharedPreferences để lưu trữ ba giá trị cấu hình sau: tắt/ bật chức năng tìm từ gần đúng, lựa chọn tùy chỉnh

chức năng khôi phục từ gốc và tùy chỉnh lựa chọn kích thước cỡ chữ hiển thị ngữ nghĩa tra từ.

5.6. Kỹ thuật phát âm từ tiếng Anh dùng API trên Android

Đối với các ngôn ngữ hòa kết như tiếng Anh. Do loại hình này có số lượng từ rất lớn (tiếng Anh khoảng 400.000 tiếng), hơn nữa tùy theo cấu trúc ngữ pháp mà cùng một từ có thể có các cách phát âm khác nhau. Vì thế ta không thể làm giống cách phát âm tiếng Việt là ghi âm đọc từng từ, mà ta phải sử dụng cách khác đó là dựa vào nguyên tắc có được khi nghiên cứu về từ vựng học. Đây là công việc kết hợp giữa ngôn ngữ học và khoa học máy tính.

Trong ứng dụng này thì ngôn ngữ phát âm là tiếng Anh, và hiện nay để phát âm được từ tiếng Anh các từ điển trên Desktop hiện nay đã sử dụng bộ SDK Text To Speech của Microsoft. Với bộ SDK này thì công việc lập trình phát âm trở nên đơn giản hơn rất nhiều. Hiện nay trên hệ điều hành Android cũng đã hỗ trợ bộ SDK này và được gọi là **Text-To-Speech API (TTS).**

Text To Speech (TTS) trên nền tảng Android bắt đầu với Android 1.4 (API Level4) đã hỗ trợ các ngôn ngữ: tiếng Anh, tiếng Pháp, tiếng Đức, tiếng Ý và tiếng Tây Ban Nha. Ngoài ra nó còn tùy thuộc vào vùng miền, như tiếng Anh giọng Mỹ và tiếng Anh giọng bản xứ đều được hỗ trợ. TTS chỉ cần biết cần đọc ngôn ngữ nào, ví dụ "Paris" có thể đọc khác nhau theo tiếng Anh và tiếng Pháp.

Mặc dù tất cả các thiết bị Android đều có hỗ trợ TTS, với một số thiết bị có bộ lưu trữ hạn chế có thể thiếu một số tập tin tài nguyên ngôn ngữ. Nếu người dùng muốn sử dụng tài nguyên này, TTS API cho phép ứng dụng truy vấn các tài nguyên có sẵn và có thể tải về cài đặt thêm. Vì vậy khi sử dụng chức năng phát âm này, bước đầu tiên là kiểm tra sự hiện diện của các tài nguyên TTS với mục đích tương ứng:

```
     Intent checkIntent = new Intent();
     checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DA
TA);
     startActivityForResult(checkIntent, DATA_CHECK_CODE);
```

Sau khi kiểm tra thành công sẽ được đánh dấu kết quả bằng một biến CHECK_VOICE_DATA_PASS và cho biết ứng dụng đã sẵn sàng tạo đối tượng TextToSpeech. Nếu không thì sẽ báo cho ứng dụng gọi Google Play tự động tải và cài đặt. Dưới đây là ví dụ về những gì thực hiện:

```
private TextToSpeech tts;

protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
     if (requestCode == DATA_CHECK_CODE) {
          if (resultCode ==
TextToSpeech.Engine.CHECK_VOICE_DATA_PASS)
               tts = new TextToSpeech(this, this);
          else {
               Intent install = new Intent();
     install.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA
);
                    startActivity(install);
               }
          }

     }
```

Trong ứng dụng này cần đọc ngôn ngữ là tiếng Anh do đó ta kiểm tra nó có hỗ trợ và sau đó cài đặt loại ngôn ngữ nhờ vào hàm *setLanguage*.

```
if (status == TextToSpeech.SUCCESS) {
if (tts.isLanguageAvailable(Locale.ENGLISH) ==
TextToSpeech.LANG_AVAILABLE)
          tts.setLanguage(Locale.ENGLISH);
     } else if (status == TextToSpeech.ERROR) {
               Toast.makeText(this, "Error when using TTS",
Toast.LENGTH_LONG).show();
}
```

Bây giờ TextToSpeech của chúng ta đã được khởi tạo và cấu hình. Chúng ta có thể bắt đầu làm cho ứng dụng phát âm một chuỗi text nhờ vào phương thức *speak*() như sau.

```
     tts.speak(text, TextToSpeech.QUEUE_ADD, null);
```

Text-To-Speech là chức năng dịch vụ được cung cấp cho nhiều ứng dụng khác trong cùng thiết bị, do đó khi xử dụng xong TTS và không cần nữa thì chúng ta nên ngừng kết nối và trả dịch vụ bằng cách gọi phương thức *shutdown()* trong Activity *onDestroy()*.

```
protected void onDestroy() {

            // Close the Text to Speech Library
            if (tts != null) {

                  tts.stop();
                  tts.shutdown();
                  // Log.d(TAG, "TTS Destroyed");
            }
            super.onDestroy();
}
```

Như vậy ứng dụng từ điển trên Android có thể dễ dàng phát âm trực tiếp được từ tiếng Anh nhờ vào Text To Speech API mà chúng ta không cần phải lưu trữ dữ liệu nhiều.

5.7. Môi trường phát triển ứng dụng

Đề tài ứng dụng tra từ điển trên camera bao gồm ứng dụng cài đặt trên điện thoại Android (tập tin cài đặt .apk) và tập tin dữ liệu đi kèm với chương trình để chép vào thẻ nhớ.

Sau đây là danh sách các công cụ và môi trường phát triển để thực hiện đề tài:

Công cụ thiết kế, vẽ mô hình, sơ đồ khối: Microsoft Word, Microsoft Visio, Paint, Photoshop.

Môi trường và công cụ lập trình:

- Eclipse IDE for Java phiên bản Indigo.
- Android SDK (cài đặt qua plug-in ADT trên Eclipse).
- Android NDK r8.

Các công cụ và thư viện sử dụng:

- Bộ cài đặt và mã nguồn của thư viện Tesseract phiên bản 3.01 trên Windows.

- Mã nguồn thư viện xử lý ảnh Leptonica 1.68 và Libjpeg.
- Công cụ tesseract-android-tools trên Eclipse.

Môi trường cài đặt và thử nghiệm:

- Máy ảo Android trên Android SDK với phiên bản Android 2.3.3 và phiên bản hàm API 14.
- Điện thoại **Sony Ecricson Neo Mt15i** chạy phiên bản Android 2.3.4.

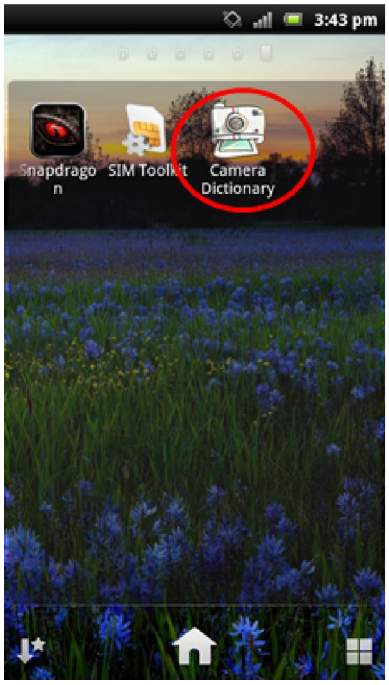
Ngoài ra còn một số chương trình khác để hỗ trợ cho ứng dụng như: phát âm tiếng Anh dùng bộ SDK Text To Speech trên Android của Microsoft…

5.8. Hướng dẫn cài đặt và sử dụng

5.8.1. Cài đặt chương trình

Chương trình **CameraDictionary** hoạt động tốt trên các loại điện thoại sử dụng nền tảng Android phiên bản 2.3 trở lên. Sau đây là cấu hình và yêu cầu cài đặt:

- Phiên bản Android Gingerbeard 2.3 trở lên.
- Điện thoại có camera và hỗ trợ tính năng tự động lấy nét (auto focus), nếu không có tính năng lấy nét này chương trình vẫn có thể sử dụng được với hình ảnh chất lượng bình thường trong điều kiện môi trường tốt.
- Bộ nhớ chính điện thoại còn trống ít nhất 20 MB cho cài đặt chương trình.
- Thẻ nhớ còn trống 38 MB để chứa các tập tin dữ liệu từ điển và dữ liệu của bộ nhận dạng Tesseract.



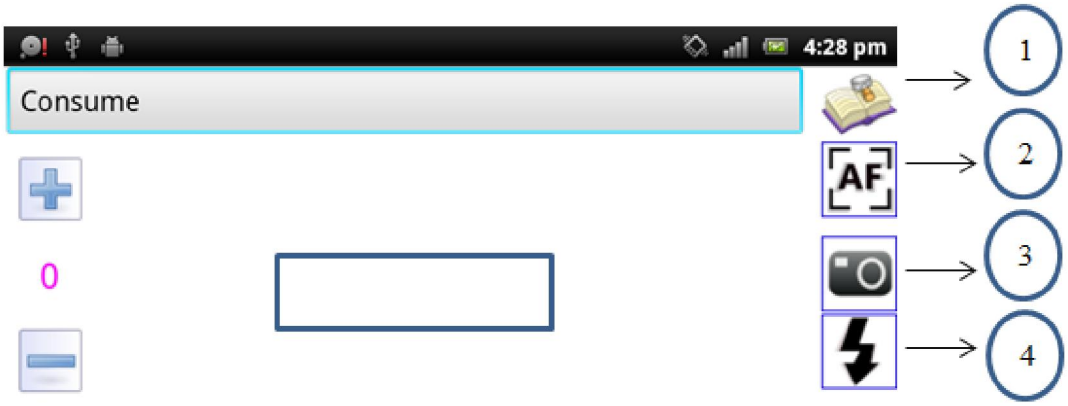
Hình 5.7 Biểu tượng chương trình sau khi cài đặt hoàn tất

Tập tin cài đặt chương trình CameraDictionary.apk và thư mục chứa dữ liệu CameraDictionary.

Đầu tiên chép tập tin cài đặt chương trình vào trong thẻ nhớ hoặc bộ nhớ điện thoại và bắt đầu cài đặt. Sau đó, chép tiếp thư mục CameraDictionary và trong thẻ nhớ.

5.8.2.Hướng dẫn sử dụng

Khởi động chương trình CameraDictionary lên và ta được giao diện màn hình chính như sau:



Hình 5.8 Màn hình chương trình khi khởi động

Cách tra từ bằng camera của chương trình: Sử dụng khung hình chữ nhật ở giữa màn hình để giới hạn lại vùng nhận diện của từ. Có thể dùng tay điều khiển kéo thả phóng to hoặc thu nhỏ vùng diện tích của khung cho vừa với từ trong vùng văn bản. Sau khi hiệu chỉnh khung giới hạn hình chữ nhật theo ý muốn, nhấn biểu tượng hình camera được đánh dấu số 3 để nhận dạng từ trong khung giới hạn. Kết quả nhận dạng từ sẽ được hiển thị ở thanh EditText bên trên màn hình. Để tra từ vừa nhận dạng xong, nhấn biểu tượng số 1 kế bên khung EditText để hiển thị màn hình nghĩa của từ.

Hình 5.9 Màn hình hiển thị nghĩa của từ

Ngoài ra để nghe phát âm từ tiếng anh trong màn hiển thị nghĩa ta nhấn biểu tượng hình cái loa ở bên góc phải màn hình và để quay về màn hình chính nhấn biểu tượng mũi tên kế bên.

Lưu ý: Để có thể sử dụng tính năng phát âm từ tiếng Anh trong chương trình, điện thoại cần phải được cài đặt hệ thống Text To Speech trên Android.

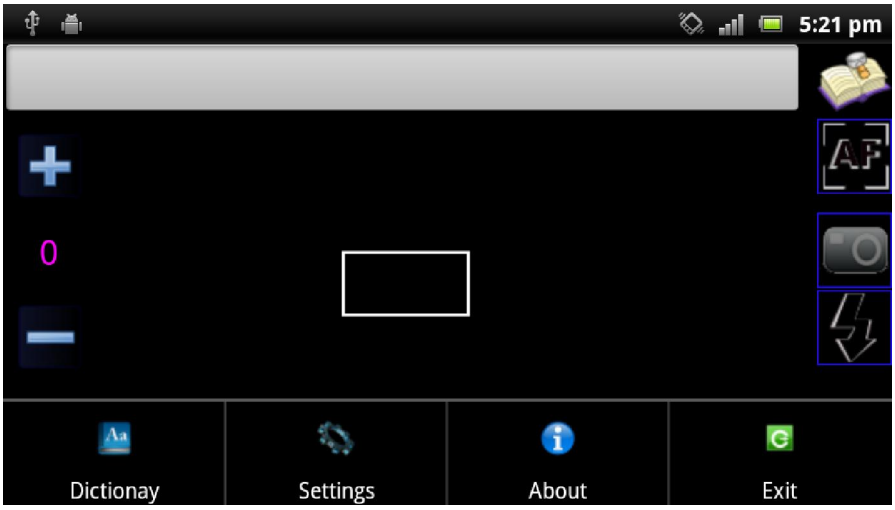
Khi người dùng nhấn biểu tượng camera để tra từ, mặc định tính năng lấy nét tự động của camera (nếu có hỗ trợ) sẽ được thực hiện để bức hình chụp sẽ được rõ nét nhất. Tuy nhiên, chương trình cũng cung cấp cho người dùng tính năng tự canh chỉnh lấy nét tự động khi người dùng cần. Sử dụng bằng cách nhấn biểu tượng số 2 trong hình. Ngoài ra, khi người dùng sử dụng để tra từ trong điều kiện thiếu ánh

sáng thì có thể nhấn biểu tượng số 4 để bật/ tắt đèn flash trong camera điện thoại (nếu điện thoại có hỗ trợ flash).

Bên góc trái của màn hình là tính năng phóng to/ thu nhỏ ảnh trên màn hình. Nếu văn bản cần nhận diện ở khoảng cách quá xa hoặc cỡ chữ nhỏ có thể sử dụng tính năng zoom này để tăng độ chính xác khi nhận diện.

Khi chữ nhận diện không chính xác, người dùng có thể chụp lại hình hoặc chỉnh sửa trực tiếp kết quả nhận diện theo ý thích.

Mục đích chính của chương trình là sử dụng camera trong điện thoại để tra từ điển, nhằm giảm bớt thời gian và khó khăn khi phải nhập từ trực tiếp. Tuy nhiên nếu người dùng muốn sử dụng tính năng tra từ điển thông thường là gõ từ và tra thì chương trình cũng cung cấp cho người dùng thực hiện điều này. Trong phần menu **setting**, chọn biểu tượng đầu tiên **dictionary** để chuyển qua tính năng tra từ thông thường.



Hình 5.10 Màn hình với hệ thống menu setting ở bên dưới

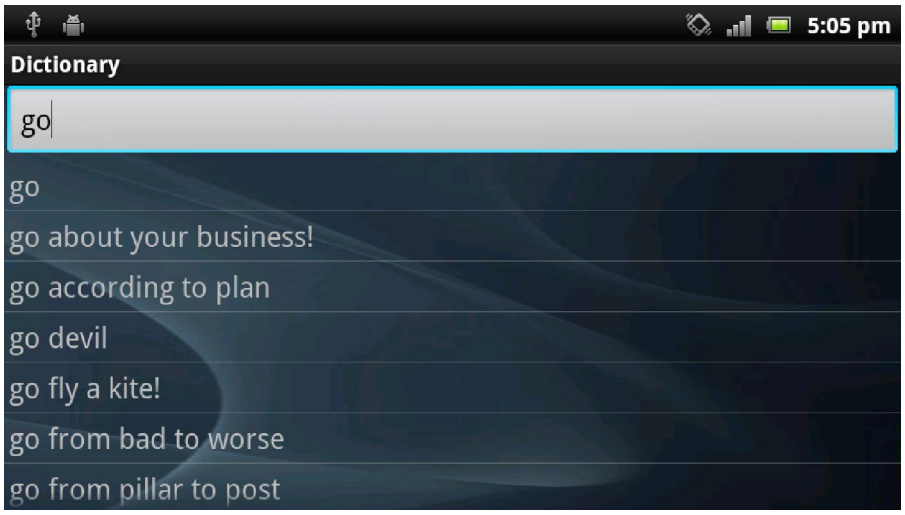
Như trong hình trên, nhấn phím setting trên điện thoại để vào hệ thống các menu con:

Dictionary: Chuyển sang tra cứu từ điển bằng cách nhập từ.

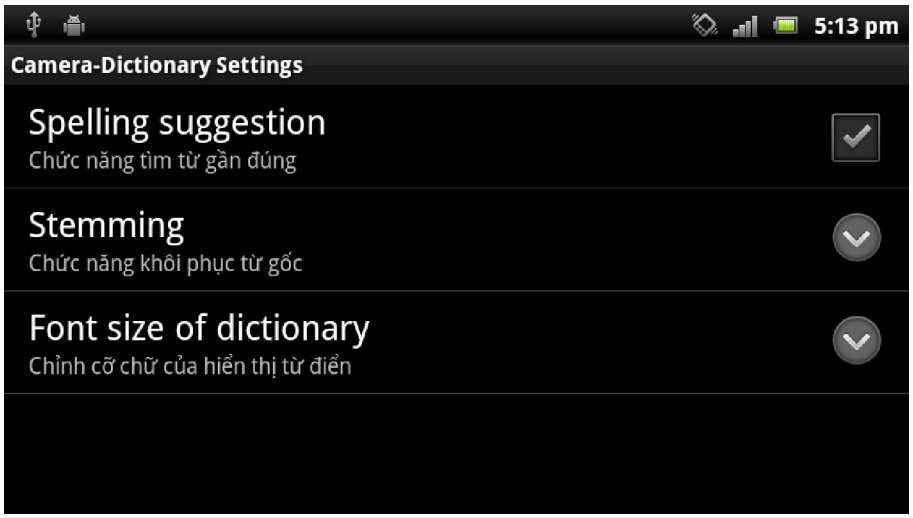
Setting: Bật tắt các tính năng nâng cao trong chương trình.

About: Hiển thị thông tin chi tiết của chương trình.

Exit: Thoát chương trình.



Hình 5.11 Màn hình tra từ điển theo cách thông thường



Hình 5.12 Màn hình thiết lập setting

- Spelling suggestion: Đây là tính năng tìm kiếm các từ gần đúng khi kết quả nhận dạng không chính xác.
- Stemming: Tính năng khôi phục từ gốc trong tiếng Anh. Việc khôi phục từ gốc sẽ được thực hiện khi tra từ.
- Font size of dictionary: Thiết lập kích thước cho font chữ hiển thị trong chương trình.

5.9. Kết quả thử nghiệm

5.9.1. Thử nghiệm khối nhận dạng ký tự

Vì dữ liệu tiếng Anh tương đối lớn mà việc thử nghiệm chỉ có thể chụp bằng tay nên nhóm chúng em chỉ chọn ra khoảng 100 từ tiếng Anh thông dụng để chụp và đánh giá kết quả. Dữ liệu được chia ra thành 2 bộ được in trên giấy trắng, mỗi bộ bao gồm 100 từ tiếng Anh. Bộ thứ 1 bao gồm 100 từ tiếng Anh sử dụng font chữ Times New Roman kích thước font chữ là 12. Bộ thứ 2 cũng bao gồm 100 từ tiếng Anh sử dụng font chữ Arial cùng kích thước 12. Cả 2 bộ đều sử dụng font chữ thường và in nghiêng.

Mặc định các ảnh được chụp thử nghiệm trong điều kiện ánh sáng tốt và đều sử dụng tính năng lấy nét tự động trong camera của điện thoại. Bảng sau đây minh họa kết quả thử nghiệm chương trình trên điện thoại Sony Neo MT15i với camera 8Mpx có hỗ trợ tự động lấy nét:

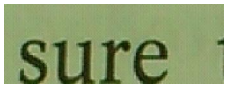
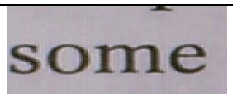
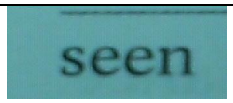
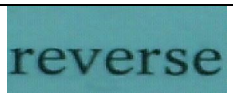
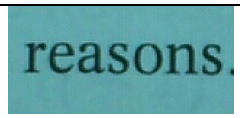
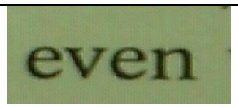
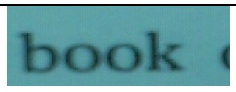
Bảng 5.1 Kết quả thử nghiệm bộ nhận dạng trong chương trình

Font chữ	Kiểu chữ	Tổng số từ	Số từ sai	Tỷ lệ lỗi
Times New Roman	In Thường	100	9	9%
Times New Roman	In nghiêng	100	16	16%
Arial	In Thường	100	12	12%
Arial	In nghiêng	100	22	22%

Dựa vào kết quả trên ta có thể thấy kết quả giữa 2 loại font Times New Roman và Arial ở kiểu chữ in thường có mức độ lỗi từ gần như nhau. Tuy nhiên, khi so sánh mức độ lỗi từ trong cùng loại font nhưng khác kiểu chữ in thường và in nghiêng thì đều chênh lệch nhau. Chữ in thường có mức độ lỗi từ thấp hơn so với chữ in nghiêng. Nhìn chung mức độ nhận dạng bị lỗi của chương trình nằm trong giới hạn dưới 20% và độ nhận dạng từ chính xác khá cao. Các lỗi này có thể khắc phục bằng cách chụp lại nhiều lần để có kết quả chính xác hơn.

Các kết quả nhận diện đúng thường được chụp trong điều kiện ánh sáng tốt và ảnh đã được lấy nét hoàn chỉnh. Các trường hợp chụp từ sai thường là do từ có nhiều nét giống kề nhau dễ lẫn lộn. Ví dụ "common" trong từ này có 2 từ "m" nằm kế nhau tạo ra nhiều nét dọc, dễ bị nhận diện sai. Hoặc các từ có 2 từ "l" nằm kế nhau. Ngoài ra các từ có thể bị nhầm lẫn với các con số hoặc các ký hiệu đặc biệt. Dưới đây là minh họa một số trường hợp nhận diện sai khi chụp thử nghiệm trên máy Sony Neo dùng Android 2.3.4:

Bảng 5.2 Một số kết quả nhận diện sai

Ảnh chụp	Kết quả nhận dạng
	SLII'€
	SOITl
	S€€l'l
	I'l€V€l
	P001'
	I'CflSO1'1S
	V€I'l
	b00k

Nhận xét: Các từ nhận diện sai thường do người dùng chụp trong điều kiện không tốt và ảnh lấy nét bị nhòe dẫn đến nhiều ký tự bị nhận diện nhầm. Trong

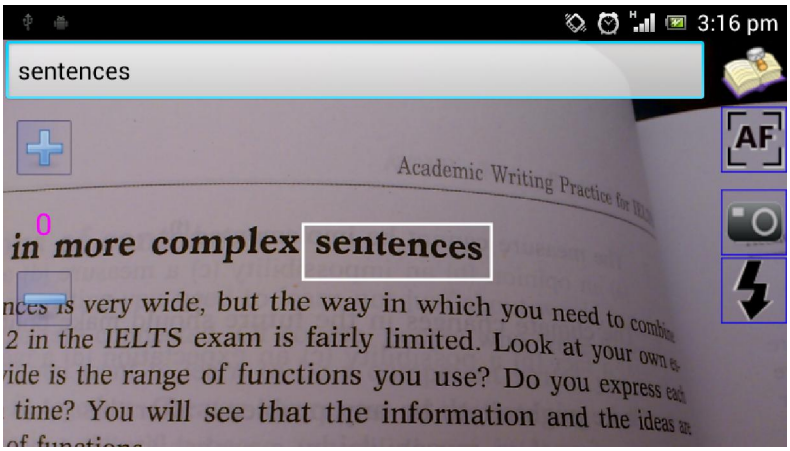
trường hợp này người dùng chỉ cần canh chỉnh camera và chụp lại ảnh để có kết quả nhận diện đúng. Dựa vào quá trình chụp thử nghiệm thì ta thấy là phần lớn các từ bị nhận diện sai đều rơi vào trường hợp các ký tự ‘e’ có trong từ. Ký tự ‘e’ dễ bị nhận nhầm thanh ký tự ‘€’. Một số trường hợp nhận nhầm khác là ký tự ‘r’ bị chuyển thành I’ hoặc 1’, ký tự ‘n’ thành I’I, ký tự ‘o’ thành ‘0’.

Trên đây là một số trường hợp ảnh bị nhận diện sai thường gặp trong chương trình. Phần lớn các lỗi liên quan đến ký tự ‘e’ đều khó sửa, các ký tự còn lại có thể chụp lại ảnh nhiều lần để cho ra kết quả nhận dạng đúng.

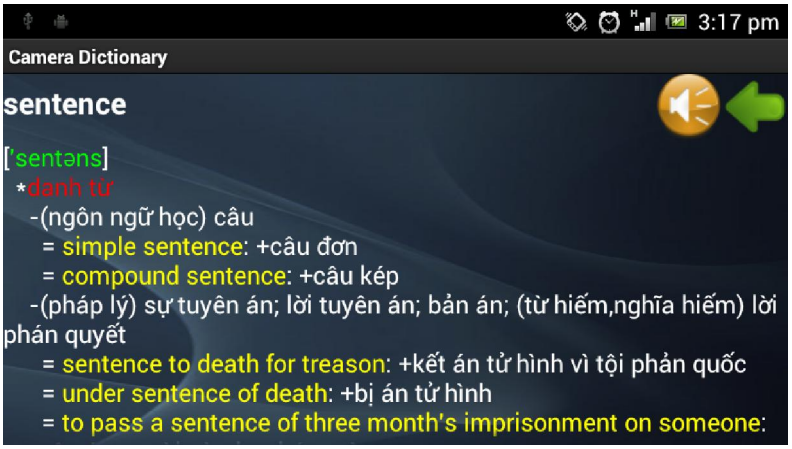
5.9.2. Thử nghiệm khối xử lý ngôn ngữ

Luận văn cũng tiến hành thử nghiệm trên khối xử lý ngôn ngữ bao gồm khối phục từ gốc và tìm từ gần đúng và sau đây là một số kết quả sau thử nghiệm.

Khôi phục từ gốc: sentences → sentence



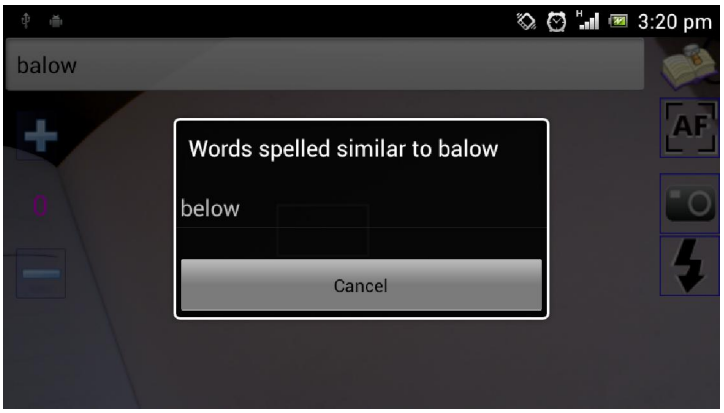
Hình 5.13 Kết quả trước khi tra từ



Hình 5.14 Màn hình hiển thị nghĩa của từ sau khi xử lý từ gốc

Nhận xét: Tính năng khôi phục từ gốc hoạt động đúng với yêu cầu đề ra và trong phần lớn các trường hợp từ nhận diện rơi vào các trường hợp thêm –ing hoặc –ed thì kết quả nhận diện sẽ được xử lý trả về từ nguyên mẫu trước khi thực hiện việc tra cứu từ điển. Tính năng khôi phục từ gốc được tích hợp sẵn khi tra từ và là tính năng tùy chọn. Tốc độ xử lý việc tra từ khi có khôi phục từ gốc tương đối nhanh và nằm trong khoảng thời gian 1 giây.

Tìm từ gần đúng: balow → below



Hình 5.15 Màn hình hiển thị danh sách từ gần đúng với kết quả nhận dạng

Nhận xét: tính năng tìm từ gần đúng hoạt động đúng với yêu cầu đề ra là liệt kê ra các danh sách từ gần với từ nhận dạng trong từ điển. Tuy nhiên, các trường

hợp liệt kê ra các từ gần đúng chỉ hoạt động hiệu quả nếu từ nhận diện chỉ có 1 ký tự bị sai, các từ có 2 hoặc 3 ký tự sai trở lên thì việc liệt kê danh sách từ gần đúng sẽ không được chính xác. Tốc độ thực thi của tính năng tìm từ gần đúng tương đối nhanh đối với các từ có ít ký tự, còn các từ chưa nhiều ký tự thì việc thực hiện sẽ chiếm thời gian lâu hơn nhưng trong khoảng thời gian chấp nhận được (khoảng 3 giây). Và đây cũng có thể xem như là 1 điểm hạn chế của tính năng tìm từ gần đúng.

5.9.3. Đánh giá kết quả

Luận văn tổng hợp lại các kết quả thực nghiệm trên máy thật Sony Neo Mt15i và đánh giá tốc độ thực thi của từng thành phần trong chương trình.

Bảng 5.3 Đánh giá tốc độ thực thi của chương trình

STT	Tính năng	Tốc độ thực thi
1	Khởi động ứng dụng	Khởi động ngay lập tức
2	Khối nhận diện từ OCR	Tương đối nhanh. Trung bình trong khoảng 3 giây
3	Tra từ điển và hiển thị nghĩa	Tra từ nhanh, hiển thị nghĩa chính xác. Tốc độ thực thi trong khoảng 1 giây
4	Tìm từ gần đúng	Tương đối nhanh, tốc độ phụ thuộc vào độ dài của từ. Trung bình trong khoảng 4 giây

5.9.4. So sánh ứng dụng với các ứng dụng hiện có trên thị trường

Đối với các loại từ điển tra từ bằng phương pháp nhập tay thông thường cho Android hiện nay thấy phổ biến nhất là Andict, sau đó là Wordmate tuy nhiên 2 từ điển này còn nhiều hạn chế, tính năng còn có phần đơn giản, Andict chỉ có dữ liệu phát âm 20.000 từ còn Wordmate không có phát âm, khi tra cứu thì chỉ trên 1 từ điển đơn, hiển thị dữ liệu trên màu nền đen gây khó chịu và không quen...

Đối với ứng dụng loại tra từ điển dùng camera trên thiết bị Android trên thị trường cũng xuất hiện 2 ứng dụng hỗ trợ tương tự là mSPDict và CamDictionary. Đối với mSPDict thì tính năng chính là tra từ điển nhập tay với dữ liệu tổng hợp từ Andict còn chức năng dùng camera nhận dạng chữ là chức năng phụ và khi nhận dạng xong thì ứng dụng tra từ qua Google Translate chứ không tra qua dữ liệu cục bộ. Ứng dụng CamDictionary do một công ty chuyên về nhận dạng văn bản từ hình ảnh nên phần nhận dạng rất tốt tuy nhiên về dịch thuật thì ứng dụng CamDictionary sử dụng từ Internet và phần phát âm cũng xử lý ở server từ Internet nên bất tiện và tốn thời gian chờ đợi. Sau đây là so sánh cụ thể một số tính năng của ứng dụng đề tài luận văn Camera Dictionary và ứng dụng trên thị trường CamDictionary.

Bảng 5.4 Bảng so sánh ứng dụng với Camera Dictionary

Tính năng so sánh	**Camera-Dictionary** (luận văn)	**CamDictionary** (thị trường)
Tập tin cài đặt	Gồm tập tin cài đặt *CameraDictionary.apk* và thư mục chứa dữ liệu từ điển *CameraDictionary*	Một tập tin cài đặt
Dung lượng	16.3 MB	18.56 MB
Bộ nhớ sử dụng khi chạy chương trình	396 KB	512 KB
Khởi động ứng dụng	1-2s hiển thị màn hình camera	1-2s hiển thị màn hình dịch đoạn văn sau đó người dùng chuyển sang màn hình camera
Tốc độ nhận dạng ảnh sang text	Trung bình 2-3s	Bộ nhận dạng xử lý rất tốt, nhanh, chính xác trung bình

		1s
Tính tiện dụng khi chụp từ	Giới hạn khung chụp, nên bất tiện hơn	Không có giới hạn khung chụp mà xử lý chữ lân cận con trỏ nên tiện lợi hơn khi sử dụng
Dữ liệu từ điển	Từ điển Anh-Việt: 106376 từ: 18,8MB	Bản Free: dữ liệu từ Google Translate. Bản License: có hỗ trợ 60% từ điển Oxford.
Tốc độ tra từ	<1s	>1s, trung bình 3-4s
Thời gian hiển thị nghĩa	Từ điển Anh-Việt tra từ "*love"* màn hình hiển thị 39 dòng với đầy đủ các phần nghĩa và các ví dụ, thời gian <1s	Từ điển Anh-Việt tra từ "*love"* màn hình hiển thị từ "*Yêu*", trong thời gian 3s.
Phát âm	Tiếng Anh dùng Text To Speech trên máy cục bộ. <1s	Dùng Internet đợi xử lý lâu >1s

TỔNG KẾT

MỘT SỐ KẾT QUẢ ĐẠT ĐƯỢC

Trong quá trình thực hiện đề tài "Tra từ điển Anh-Việt qua camera trên điện thoại di động dùng Android" chúng em thu được những kết quả sau:

- Tìm hiểu được các loại điện thoại di động và kỹ thuật lập trình ứng dụng trên thiết bị di động.
- Tìm hiểu được hệ điều hành Android và các thiết bị sử dụng hệ điều hành này. Nền tảng Android của Google giữ vị trí số 1, nhờ vào số lượng phiên bản phong phú và nhận được sự hỗ trợ từ hầu hết các hãng sản xuất phần cứng lớn.
- Tìm hiểu được cách tổ chức dữ liệu từ điển hiệu quả trong môi trường di động có nhiều hạn chế về tài nguyên. Và tìm hiểu phương pháp mã hóa, giải mã dữ liệu từ điển.
- Xây dựng thành công ứng dụng Camera-Dictionary trên điện thoại di động dùng Android 2.3 trở lên có hỗ trợ camera. Với dữ liệu phong phú, tra từ nhanh và chính xác, hoạt động ổn định. Ứng dụng có hỗ trợ phát âm tiếng Anh giúp người dùng có thể học dễ dàng. Camera-Dictionary đã phần nào đáp ứng nhu cầu và góp phần mang lại phương thức tra từ tiện lợi cho người dùng. Ứng dụng Camera-Dictionay có các tính năng sau:

Bảng 5.5 Các tính năng chính trong chương trình

STT	Tên tính năng
1	Tra từ điển trực tiếp bằng camera điện thoại
2	Bật tắt đèn flash khi cần làm sáng môi trường chụp
3	Tự động lấy nét ảnh.
4	Phóng to, thu nhỏ ống kính camera
5	Tra từ điển bằng cách nhập từ bàn phím

6	Hiển thị danh sách các từ kế tiếp trong từ điển
7	Khôi phục từ gốc, đưa về nguyên mẫu các trường hợp thêm -s, -es, -ed, -ing…
8	Tìm từ gần đúng, hiển thị danh sách các từ gợi ý liên quan khi từ cần tra không có trong từ điển
9	Thay đổi font chữ hiển thị nghĩa của từ
10	Đọc, phát âm từ tiếng Anh

HẠN CHẾ

➢ Ứng dụng chỉ xử lý ảnh tốt hơn khi camera điện thoại có hỗ trợ lấy nét tự động (auto focus). Thiết bị camera có độ phân giải càng cao thì ảnh càng rõ nét và bộ nhận dạng ký tự quang học làm việc càng hiệu quả.

➢ Hiệu quả chụp ảnh tra từ còn phụ thuộc vào ánh sáng môi trường và điều kiện màu nền, phông chữ phù hợp. Khi ứng dụng chụp môi trường chữ trên các vật dụng bên ngoài thì kích thước chữ to giúp nhận dạng chính xác hơn.

➢ Thuật toán tìm từ gần đúng còn hạn chế do chỉ xử lý trường hợp sai một ký tự, chưa xử lý các trường hợp sai cùng lúc nhiều ký tự. Do tốc độ xử lý của máy cục bộ còn hạn chế.

HƯỚNG PHÁT TRIỂN

➢ Bổ sung hoàn chỉnh bộ dữ liệu từ điển, nhất là từ điển chuyên môn.

➢ Bổ sung bộ nhận dạng ký tự quang học và dữ liệu từ điển, nhất là đối với các ngôn ngữ không sử dụng mẫu tự Latinh như Hoa, Nhật, Hàn, Thái,… Việc này mang ý nghĩa quan trọng vì các ngôn ngữ này gây khó khăn cho từ điển thông thường không thể nhập từ cần tra từ bàn phím thiết bị di động cho người mới sử dụng.

➢ Cải thiện xử lý ảnh để ảnh chụp được rõ nét trong nhiều điều kiện ánh sáng khác nhau và trong nhiều nguồn chụp khác nhau. Việc này hỗ trợ

người dùng có thể chụp tra từ ở mọi nơi, nhất là đối với du khách du lịch có thể chụp tra từ các bảng hiệu, chỉ dẫn…

- Cải thiện khối nhận dạng quang học nhận dạng nhiều ký tự hơn không cần giới hạn vùng chụp và ứng dụng hỗ trợ dịch cụm từ, dịch câu bằng camera điện thoại.
- Xây dựng thêm tính năng tra từ trực tuyến dùng Google Translate thông qua Internet (GPRS, 3G) khi ứng dụng tra các từ không tìm thấy dữ liệu trên máy cục bộ.

TÀI LIỆU THAM KHẢO

Tài liệu viết:

[1] Bùi Tấn Lộc, Cao Thái Phương Thanh, *Nghiên cứu và xây dựng ứng dụng từ điển trên điện thoại di động*, Luận văn cử nhân tin học, Đại học Khoa học Tự nhiên TP.HCM, 2004.

[2] Nguyễn Hoàng Giang, *Xây dựng ứng dụng tra từ điển bằng camera trên điện thoại di động*, Luận văn thạc sĩ tin học, Đại học Khoa học Tự Nhiên TP.HCM, 2011.

[3] Marko Gargenta, *Learning Android*, O'REILLY, 2011.

[4] Jurij Smakov, *JNI Examples for Android,* 2009.

[5] Ray Smith, *An overview of Tesseract OCR engine*, 2007.

[6] Ray Smith, Daria Antonova, Dar Shyang-Lee, *Adapting the Tesseract Open Source OCR Engine for Multilingual OCR,* ACM, 2009.

Website:

[1] http://developer.android.com/index.html

[2] http://tesseract-ocr.googlecode.com/files/TesseractOSCON.pdf

[3] http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract

[4] http://tartarus.org/~martin/PorterStemmer/

[5] http://en.wikipedia.org/wiki/Data_Encryption_Standard

[6] http://www.java2s.com/Code/Java/Security/DES.htm

[7] http://en.wikipedia.org/wiki/Levenshtein_distance