ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG
----------o0o----------

ĐỒ ÁN MÔN HỌC

TÓM TẮT EMAIL THÔNG MINH

GVHD: TS. PHẠM QUANG THÁI
SVTH: NGUYỄN CÔNG HÙNG
MSSV: 1913606

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2025

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến TS. Phạm Quang Thái - giảng viên hướng dẫn đã tận tình chỉ bảo, hỗ trợ và đóng góp nhiều ý kiến quý báu giúp em hoàn thành đồ án này.

Em cũng xin cảm ơn Khoa Điện - Điện tử, Bộ môn Điện tử, Trường Đại học Bách Khoa TP.HCM đã tạo điều kiện tốt nhất cho em trong quá trình học tập và nghiên cứu.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, bạn bè đã luôn bên cạnh, động viên và hỗ trợ em trong suốt quá trình thực hiện đề tài này.

Em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, ngày 29 tháng 5 năm 2025 .

Sinh viên

Nguyễn Công Hùng

TÓM TẮT ĐỒ ÁN

Đồ án này trình bày về việc xây dựng ứng dụng " Tóm tắt Email thông minh" sử dụng ngôn ngữ Python. Ứng dụng có khả năng kết nối với tài khoản Gmail của người dùng, truy xuất và hiển thị danh sách email, đồng thời cung cấp tính năng tóm tắt nội dung email bằng kỹ thuật xử lý ngôn ngữ tự nhiên. Ứng dụng hỗ trợ đặc biệt cho tiếng Việt với thuật toán tóm tắt văn bản được tối ưu hóa cho đặc thù của ngôn ngữ này. Giao diện người dùng được xây dựng bằng thư viện Tkinter, đảm bảo sử dụng đơn giản, trực quan và thân thiện. Kết quả thử nghiệm cho thấy ứng dụng có khả năng xử lý hiệu quả các email có độ dài và nội dung khác nhau, cung cấp bản tóm tắt ngắn gọn, giúp người dùng nhanh chóng nắm bắt thông tin quan trọng mà không cần đọc toàn bộ nội dung.

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH BẢNG SỐ LIỆU

1. GIỚI THIỆU

1.1 Tổng quan

Trong thời đại số hóa hiện nay, email đã trở thành một phương tiện giao tiếp không thể thiếu trong công việc và cuộc sống hàng ngày. Tuy nhiên, với khối lượng email ngày càng tăng, người dùng thường gặp khó khăn trong việc quản lý và xử lý thông tin. Đặc biệt, nhiều email có nội dung dài dòng, khiến người đọc mất nhiều thời gian để nắm bắt thông tin quan trọng.

Trong bối cảnh đó, các giải pháp tự động hóa việc phân tích và tóm tắt email trở nên vô cùng hữu ích. Các công nghệ xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đã phát triển đến mức có thể ứng dụng vào việc tóm tắt văn bản với độ chính xác cao, giúp người dùng tiết kiệm thời gian đáng kể trong việc đọc và xử lý email.

Đối với tiếng Việt, công nghệ xử lý ngôn ngữ tự nhiên còn đang trong giai đoạn phát triển so với các ngôn ngữ phổ biến như tiếng Anh. Vì vậy, việc nghiên cứu và phát triển một ứng dụng tóm tắt email hỗ trợ tiếng Việt là một đề tài có ý nghĩa, đáp ứng nhu cầu thực tế của người dùng Việt Nam.

1.2 Nhiệm vụ đề tài

Đề tài này tập trung vào việc phát triển một ứng dụng phân tích và tóm tắt email với các nhiệm vụ cụ thể như sau:

1.2.1 Tìm hiểu và nghiên cứu lý thuyết:

- Tìm hiểu về cách thức kết nối và truy xuất dữ liệu từ Gmail API
- Nghiên cứu các phương pháp xử lý ngôn ngữ tự nhiên và tóm tắt văn bản
- Tìm hiểu đặc điểm của ngôn ngữ tiếng Việt và phương pháp tối ưu cho việc tóm tắt

1.2.2 Xây dựng thuật toán tóm tắt email:

- Thiết kế thuật toán trích xuất các câu quan trọng từ nội dung email

- Phát triển phương pháp tóm tắt phù hợp với đặc thù của tiếng Việt
- Tối ưu hóa thuật toán cho hiệu năng cao

1.2.3 Phát triển ứng dụng desktop:

- Xây dựng giao diện người dùng trực quan, dễ sử dụng
- Tích hợp tính năng xác thực và kết nối với Gmail
- Hiển thị danh sách email và cho phép người dùng xem nội dung cũng như bản tóm tắt

1.2.4 Đánh giá và tối ưu hóa:

- Thử nghiệm trên nhiều loại email với nội dung và độ dài khác nhau
- Đánh giá độ chính xác và hiệu quả của thuật toán tóm tắt
- Tối ưu hóa hiệu suất tổng thể của ứng dụng

2. LÝ THUYẾT

2.1 Kết nối và truy xuất email từ Gmail

Để kết nối và truy xuất dữ liệu từ Gmail, ứng dụng sử dụng Gmail API (Application Programming Interface) được cung cấp bởi Google. Gmail API cho phép các ứng dụng bên thứ ba truy cập và quản lý dữ liệu email của người dùng một cách an toàn thông qua xác thực OAuth 2.0.

2.1.1 Xác thực OAuth 2.0

OAuth 2.0 là một giao thức xác thực cho phép các ứng dụng bên thứ ba truy cập tài nguyên của người dùng mà không cần biết thông tin đăng nhập của họ. Quy trình xác thực gồm các bước chính:

1. Đăng ký ứng dụng trên Google Developer Console để nhận Client ID và Client Secret
2. Yêu cầu người dùng cấp quyền truy cập vào dữ liệu Gmail của họ

3. Nhận mã xác thực (authorization code) từ Google
4. Đổi mã xác thực lấy access token và refresh token
5. Sử dụng access token để truy cập API

Trong ứng dụng, thư viện google-auth-oauthlib được sử dụng để triển khai quy trình xác thực OAuth 2.0. Access token và refresh token được lưu trữ trong file token.json để sử dụng cho các lần truy cập tiếp theo.

2.1.2 Truy xuất và xử lý dữ liệu email

Gmail API cung cấp nhiều phương thức để truy xuất và quản lý email. Trong ứng dụng này, các phương thức chính được sử dụng bao gồm:

- users().messages().list(): Lấy danh sách ID của các email
- users().messages().get(): Lấy nội dung chi tiết của một email dựa trên ID

Dữ liệu email từ Gmail API được trả về dưới dạng JSON với cấu trúc phức tạp. Email bao gồm các phần chính:

- Headers: Chứa thông tin về người gửi, người nhận, tiêu đề, ngày gửi
- Body: Chứa nội dung email, có thể ở dạng text/plain hoặc text/html
- Attachments: Tệp đính kèm (nếu có)

Nội dung của email thường được mã hóa base64, cần được giải mã trước khi xử lý. Nếu nội dung ở dạng HTML, cần chuyển đổi sang text để thuận tiện cho việc tóm tắt.

2.2 Xử lý ngôn ngữ tự nhiên và tóm tắt văn bản

2.2.1 Tổng quan về tóm tắt văn bản tự động

Tóm tắt văn bản tự động (Automatic Text Summarization) là quá trình tạo ra một phiên bản ngắn gọn của văn bản gốc mà vẫn giữ được các thông tin quan trọng. Có hai phương pháp chính:

1. **Tóm tắt trích xuất (Extractive Summarization)**: Chọn lọc và trích xuất các câu quan trọng nhất từ văn bản gốc để tạo thành bản tóm tắt.

2. **Tóm tắt trừu tượng (Abstractive Summarization)**: Tạo ra các câu mới, cô đọng ý chính của văn bản gốc, tương tự như cách con người tóm tắt.

Trong ứng dụng này, phương pháp tóm tắt trích xuất được sử dụng vì tính đơn giản và hiệu quả cao cho ứng dụng thực tế.

2.2.2 Thuật toán tóm tắt trích xuất

Thuật toán tóm tắt trích xuất trong ứng dụng dựa trên phương pháp tính điểm cho từng câu dựa trên tần suất xuất hiện của các từ quan trọng. Các bước chính của thuật toán bao gồm:

1. **Tiền xử lý văn bản**: Loại bỏ các ký tự đặc biệt, URL, và chuẩn hóa văn bản
2. **Phân tách câu**: Chia văn bản thành các câu riêng biệt
3. **Phân tách từ**: Chia mỗi câu thành các từ (tokens)
4. **Loại bỏ stopwords**: Loại bỏ các từ phổ biến không mang nhiều ý nghĩa
5. **Tính trọng số các từ**: Tính tần suất xuất hiện của mỗi từ trong văn bản
6. **Tính điểm cho mỗi câu**: Tổng hợp trọng số của các từ trong câu
7. **Chọn câu quan trọng nhất**: Chọn N câu có điểm cao nhất
8. **Sắp xếp theo thứ tự ban đầu**: Sắp xếp các câu được chọn theo thứ tự xuất hiện trong văn bản gốc

2.2.3 Đặc thù của tiếng Việt trong xử lý ngôn ngữ tự nhiên

Tiếng Việt có những đặc điểm riêng ảnh hưởng đến việc xử lý ngôn ngữ tự nhiên:

1. **Tính chất đơn lập**: Tiếng Việt là ngôn ngữ đơn lập, mỗi từ không biến đổi hình thái theo ngữ pháp
2. **Dấu câu và kết thúc câu**: Cách nhận diện kết thúc câu phức tạp hơn do đặc thù sử dụng dấu
3. **Từ ghép và từ láy**: Tiếng Việt có nhiều từ ghép và từ láy cần xử lý đặc biệt
4. **Dấu thanh**: Các dấu thanh ảnh hưởng đến ý nghĩa của từ

Để xử lý hiệu quả tiếng Việt, ứng dụng sử dụng thư viện underthesea - một thư viện NLP chuyên biệt cho tiếng Việt, hỗ trợ các tác vụ như tách từ (word tokenization), gán nhãn từ loại (POS tagging), và nhận diện thực thể có tên (NER).

2.3 Giao diện người dùng với Tkinter

2.3.1 Tổng quan về Tkinter

Tkinter là thư viện giao diện người dùng đồ họa (GUI) tiêu chuẩn của Python, được tích hợp sẵn trong bản cài đặt Python. Tkinter dựa trên thư viện Tk, một công cụ GUI đa nền tảng phổ biến.

Tkinter cung cấp nhiều widget (thành phần giao diện) cơ bản như:

- Frame: Khung chứa các widget khác
- Label: Hiển thị văn bản hoặc hình ảnh
- Button: Nút bấm
- Entry: Trường nhập liệu đơn dòng
- Text/ScrolledText: Vùng văn bản đa dòng có thanh cuộn
- Treeview: Hiển thị dữ liệu dạng bảng hoặc cây

2.3.2 Thiết kế giao diện người dùng

Giao diện người dùng của ứng dụng được thiết kế theo nguyên tắc đơn giản, trực quan và dễ sử dụng. Cấu trúc chính của giao diện bao gồm:

1. **Thanh tiêu đề**: Hiển thị tên ứng dụng
2. **Thanh công cụ**: Chứa các nút chức năng và điều khiển
    - Nút "Kết nối Gmail": Thực hiện xác thực và kết nối với tài khoản Gmail
    - Nút "Phân tích": Tiến hành truy xuất và phân tích email
    - Điều khiển số lượng email: Cho phép chọn số lượng email cần phân tích
    - Điều khiển số câu tóm tắt: Cho phép chọn số câu trong bản tóm tắt

3. **Vùng hiển thị danh sách email**: Hiển thị các email dưới dạng bảng với các cột STT, Người gửi, Tiêu đề, Ngày
4. **Vùng hiển thị nội dung email**: Hiển thị nội dung đầy đủ của email được chọn
5. **Vùng hiển thị tóm tắt**: Hiển thị bản tóm tắt của email được chọn
6. **Thanh trạng thái**: Hiển thị thông tin về trạng thái hiện tại của ứng dụng

2.3.3 Mô hình sự kiện và luồng dữ liệu

Tkinter sử dụng mô hình sự kiện (event-driven) trong đó các widget phát sinh sự kiện khi người dùng tương tác, và các hàm gọi lại (callback) sẽ được kích hoạt để xử lý các sự kiện đó.

Trong ứng dụng, các sự kiện chính bao gồm:

- Nhấn nút "Kết nối Gmail": Kích hoạt quá trình xác thực OAuth 2.0
- Nhấn nút "Phân tích": Bắt đầu quá trình truy xuất và phân tích email
- Chọn một email trong danh sách: Hiển thị nội dung và tóm tắt của email được chọn

Để đảm bảo giao diện vẫn phản hồi trong quá trình xử lý dài, các tác vụ nặng như xác thực, truy xuất và phân tích email được thực hiện trong các luồng (thread) riêng biệt.

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

3.1 Yêu cầu đặt ra cho phần mềm

3.1.1 Yêu cầu chức năng

1. **Xác thực và kết nối Gmail**
    - Ứng dụng phải hỗ trợ xác thực OAuth 2.0 với Gmail API
    - Lưu thông tin xác thực để sử dụng cho các lần truy cập tiếp theo
    - Hiển thị thông báo thành công/thất bại khi kết nối
2. **Truy xuất và hiển thị email**
    - Cho phép người dùng chọn số lượng email cần truy xuất (từ 1 đến 50)
    - Hiển thị danh sách email bao gồm thông tin: STT, Người gửi, Tiêu đề, Ngày
    - Hiển thị nội dung đầy đủ của email khi người dùng chọn
3. **Tóm tắt nội dung email**
    - Cho phép người dùng chọn số câu trong bản tóm tắt (từ 1 đến 10)
    - Hỗ trợ tóm tắt email bằng tiếng Việt
    - Hiển thị bản tóm tắt khi người dùng chọn email
4. **Giao diện người dùng**
    - Giao diện trực quan, dễ sử dụng
    - Hiển thị tiến trình khi thực hiện các tác vụ dài
    - Hỗ trợ chức năng sao chép nội dung

3.1.2 Yêu cầu phi chức năng

1. **Hiệu năng**
    - Thời gian phản hồi khi chọn email không quá 2 giây
    - Thời gian tóm tắt email không quá 5 giây cho email có độ dài trung bình

- Xử lý được email có độ dài lên đến 100,000 ký tự

2. **Bảo mật**
   - Bảo vệ thông tin xác thực của người dùng
   - Không lưu trữ nội dung email trên thiết bị
   - Tuân thủ các quy định về quyền riêng tư của Google
3. **Độ tin cậy**
   - Ứng dụng hoạt động ổn định, không bị treo hoặc crash
   - Xử lý lỗi và hiển thị thông báo phù hợp
   - Khả năng phục hồi sau khi mất kết nối internet
4. **Khả năng mở rộng**
   - Thiết kế mô-đun, dễ dàng thêm tính năng mới
   - Hỗ trợ tích hợp với các dịch vụ email khác trong tương lai
   - Khả năng cải tiến thuật toán tóm tắt mà không ảnh hưởng đến kiến trúc tổng thể

3.2 Phân tích

3.2.1 Phương pháp kết nối Gmail

Có hai phương pháp chính để kết nối và truy xuất dữ liệu từ Gmail:

1. **Sử dụng Gmail API**
   - Ưu điểm:
     - Bảo mật cao với xác thực OAuth 2.0
     - Truy cập chi tiết đến các thành phần của email
     - Hỗ trợ tốt từ Google và cộng đồng
   - Nhược điểm:
     - Cần đăng ký ứng dụng trên Google Developer Console

- Quá trình xác thực phức tạp hơn

2. **Sử dụng IMAP (Internet Message Access Protocol)**
    - Ưu điểm:
        - Triển khai đơn giản hơn
        - Hoạt động với nhiều dịch vụ email khác nhau
    - Nhược điểm:
        - Yêu cầu lưu trữ mật khẩu người dùng
        - Hạn chế trong việc truy cập các thành phần chi tiết của email
        - Ít bảo mật hơn OAuth 2.0

Sau khi phân tích, phương pháp sử dụng Gmail API được chọn vì tính bảo mật cao và khả năng truy cập chi tiết đến các thành phần của email.

3.2.2 Phương pháp tóm tắt văn bản

Có hai phương pháp chính để tóm tắt nội dung email:

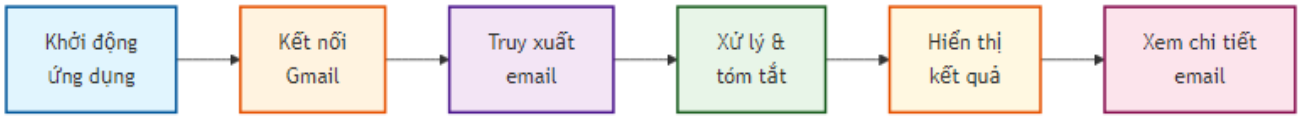
1. **Tóm tắt trích xuất (Extractive Summarization)**
    - Ưu điểm:
        - Đơn giản về mặt thuật toán
        - Hiệu quả cho văn bản có cấu trúc tốt
        - Dễ triển khai và không yêu cầu tài nguyên cao
    - Nhược điểm:
        - Không tạo ra nội dung mới
        - Có thể còn dư thừa thông tin
        - Phụ thuộc nhiều vào chất lượng văn bản gốc

2. **Tóm tắt trừu tượng (Abstractive Summarization)**
    - Ưu điểm:
        - Tạo ra nội dung ngắn gọn và mạch lạc hơn
        - Khả năng tổng hợp thông tin từ nhiều phần của văn bản
        - Bản tóm tắt giống với cách tóm tắt của con người
    - Nhược điểm:
        - Yêu cầu mô hình phức tạp như mạng neural sâu
        - Cần nhiều dữ liệu huấn luyện
        - Tài nguyên tính toán cao
        - Khó triển khai cho tiếng Việt do hạn chế về dữ liệu

Sau khi phân tích, phương pháp tóm tắt trích xuất được chọn vì tính thực tiễn và hiệu quả cho ứng dụng desktop, đặc biệt là khi xử lý tiếng Việt. Thuật toán được điều chỉnh để phù hợp với đặc thù của tiếng Việt.

3.3 Lưu đồ giải thuật chi tiết

3.3.1 Lưu đồ tổng quan của ứng dụng



Hình 3. 1 Lưu đồ tổng quan của ứng dụng

Hình 3.1 mô tả lưu đồ tổng quan của ứng dụng " Tóm tắt Email", bao gồm các quy trình chính:

1. Khởi tạo ứng dụng
2. Kết nối Gmail
3. Truy xuất danh sách email
4. Hiển thị danh sách email
5. Xem và tóm tắt email được chọn

3.3.2 Lưu đồ thuật toán tóm tắt tiếng Việt

Hình 3. 2 Lưu đồ chi tiết thuật toán tóm tắt tiếng Việt

Hình 3-2 mô tả chi tiết thuật toán tóm tắt văn bản tiếng Việt. Các bước chính bao gồm:

1. Tiền xử lý văn bản:
    - Loại bỏ các ký tự đặc biệt, URL, email
    - Chuẩn hóa khoảng trắng và dấu câu
    - Loại bỏ các phần không liên quan (ví dụ: chữ ký, disclaimers)
2. Phân tách câu:
    - Sử dụng biểu thức chính quy để nhận diện kết thúc câu trong tiếng Việt
    - Xử lý các trường hợp đặc biệt (ví dụ: viết tắt, số, v.v.)
3. Phân tách từ và loại bỏ stopwords:
    - Sử dụng underthesea để tách từ trong tiếng Việt
    - Loại bỏ các từ không mang nhiều ý nghĩa (stopwords)
4. Tính điểm cho từng câu:
    - Tính tần suất xuất hiện của mỗi từ
    - Tính tổng điểm cho mỗi câu dựa trên tần suất từ
    - Chuẩn hóa điểm theo độ dài câu
5. Chọn câu quan trọng nhất:
    - Sử dụng thuật toán heap để chọn N câu có điểm cao nhất
    - Sắp xếp các câu theo thứ tự ban đầu trong văn bản

3.4 Cấu trúc chương trình

Chương trình được tổ chức theo kiến trúc hướng đối tượng với các lớp chính sau:

1. **Constants**: Chứa các hằng số và thông số cấu hình của ứng dụng
2. **ResourceLoader**: Chịu trách nhiệm tải các tài nguyên cần thiết
3. **VietnameseTextSummarizer**: Triển khai thuật toán tóm tắt văn bản tiếng Việt
4. **GmailAnalyzer**: Xử lý kết nối và truy xuất dữ liệu từ Gmail
5. **EmailAnalyzerGUI**: Triển khai giao diện người dùng đồ họa

Ứng dụng được thiết kế theo mô hình MVC (Model-View-Controller):

- **Model**: GmailAnalyzer và VietnameseTextSummarizer
- **View**: EmailAnalyzerGUI
- **Controller**: Các phương thức xử lý sự kiện trong EmailAnalyzerGUI

3.5 Triển khai chi tiết

3.5.1 Lớp Constants

Lớp Constants định nghĩa các hằng số và thông số cấu hình của ứng dụng, bao gồm:

- Cài đặt giao diện: kích thước font, kích thước cửa sổ, độ rộng cột
- Giá trị mặc định: số lượng email, số câu tóm tắt
- Màu sắc: màu chủ đạo, màu nền, màu văn bản

class Constants:

FONT_SIZE_NORMAL = 10

FONT_SIZE_BUTTON = 11

FONT_SIZE_TITLE = 20

DEFAULT_EMAIL_COUNT = 10

```
MAX_EMAIL_COUNT = 50

MIN_EMAIL_COUNT = 1

DEFAULT_SUMMARY_SENTENCES = 3

MAX_SUMMARY_SENTENCES = 10

MIN_SUMMARY_SENTENCES = 1

# Các hằng số khác...
```

3.5.2 Lớp VietnameseTextSummarizer

Lớp VietnameseTextSummarizer triển khai thuật toán tóm tắt văn bản tiếng Việt với các phương thức chính:

```
class VietnameseTextSummarizer:

    def __init__(self):

        # Khởi tạo danh sách stopwords tiếng Việt

        self.vietnamese_stop_words = {...}

        self.stop_words = self.vietnamese_stop_words.union(set(punctuation))

        def preprocess_text(self, text):

        # Tiền xử lý văn bản: loại bỏ URL, email, chuẩn hóa khoảng trắng

        def split_sentences(self, text):

        # Phân tách văn bản thành các câu

       def split_words(self, sentence):

        # Tách từ trong câu và loại bỏ stopwords

      def summarize(self, text, num_sentences=3):

        # Tóm tắt văn bản với số lượng câu chỉ định
```

3.5.3 Lớp GmailAnalyzer

Lớp GmailAnalyzer xử lý kết nối với Gmail API và truy xuất dữ liệu email:

```
class GmailAnalyzer:
    def __init__(self):
        self.SCOPES = ['https://www.googleapis.com/auth/gmail.readonly']
        self.service = None
        self.summarizer = VietnameseTextSummarizer()

    def connect(self):
        # Xác thực và kết nối với Gmail API

    def get_emails(self, max_results=10):
        # Lấy danh sách email từ hộp thư đến

    def decode_email_content(self, payload):
        # Giải mã nội dung email

    def get_email_content(self, msg_id, summary_sentences=3):
        # Lấy nội dung chi tiết của email và tóm tắt
```

3.5.4 Lớp EmailAnalyzerGUI

Lớp EmailAnalyzerGUI triển khai giao diện người dùng đồ họa và xử lý tương tác:

```
class EmailAnalyzerGUI:
    def __init__(self):
        # Khởi tạo giao diện chính
    def async_init(self):
        # Khởi tạo không đồng bộ các tài nguyên
    def setup_styles(self):
        # Thiết lập các style cho giao diện
    def setup_gui(self):
        # Thiết lập các thành phần giao diện
    def connect_gmail(self):
        # Xử lý kết nối Gmail
    def start_analysis(self):
        # Bắt đầu phân tích email
    def on_select_email(self, event):
        # Xử lý sự kiện chọn email
    # Các phương thức khác...
```

4. KẾT QUẢ THỰC HIỆN

4.1 Môi trường thực nghiệm

Ứng dụng " Tóm tắt Email" đã được thử nghiệm trên các máy tính với cấu hình khác nhau. Bảng 4-1 trình bày thông số kỹ thuật của máy tính chính được sử dụng cho việc thử nghiệm.

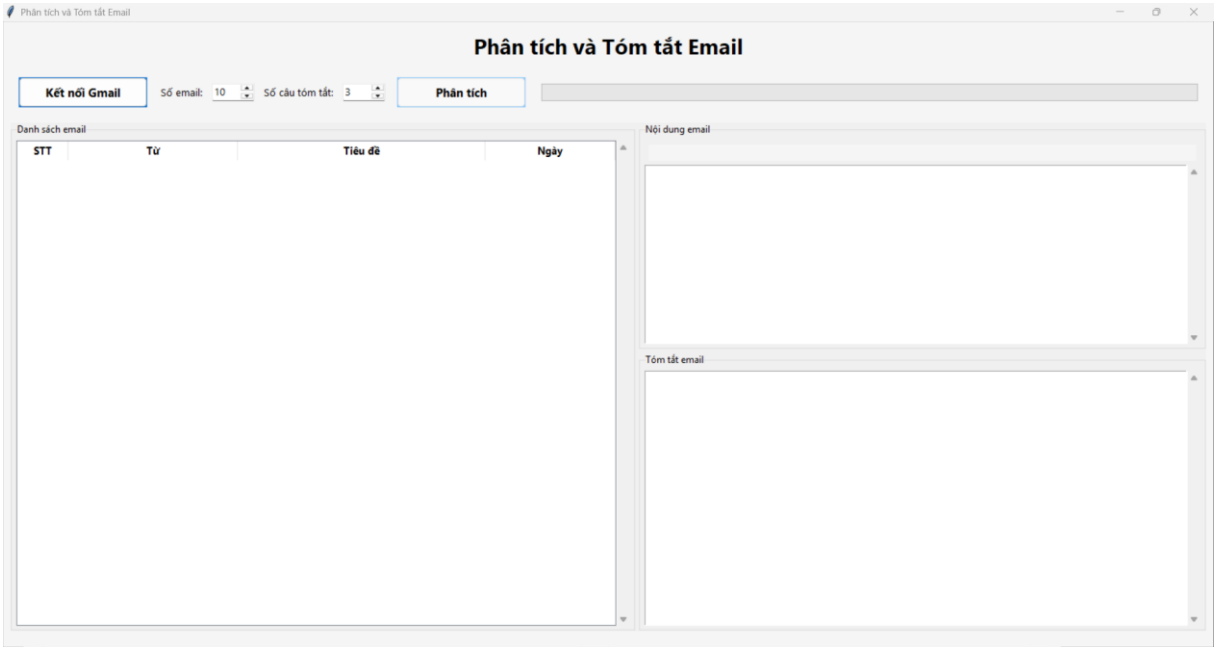
Thông số	**Cấu hình**
CPU	Intel Core i7-11800, 2.3 GHz
RAM	16GB
Hệ điều hành	Windows 11 Pro 64-bit
Python	Python 3.9.10
Màn hình	15.6 inch
Kết nối internet	Fiber 100Mbps

Bảng 4. 1 Thông số kỹ thuật máy tính thử nghiệm

Các thư viện chính được sử dụng trong dự án:

- tkinter: Giao diện người dùng đồ họa
- google-auth-oauthlib, google-api-python-client: Kết nối Gmail API
- beautifulsoup4: Xử lý nội dung HTML
- nltk: Xử lý ngôn ngữ tự nhiên
- underthesea: Xử lý ngôn ngữ tiếng Việt
- threading, concurrent.futures: Lập trình đa luồng

4.2 Kết quả giao diện



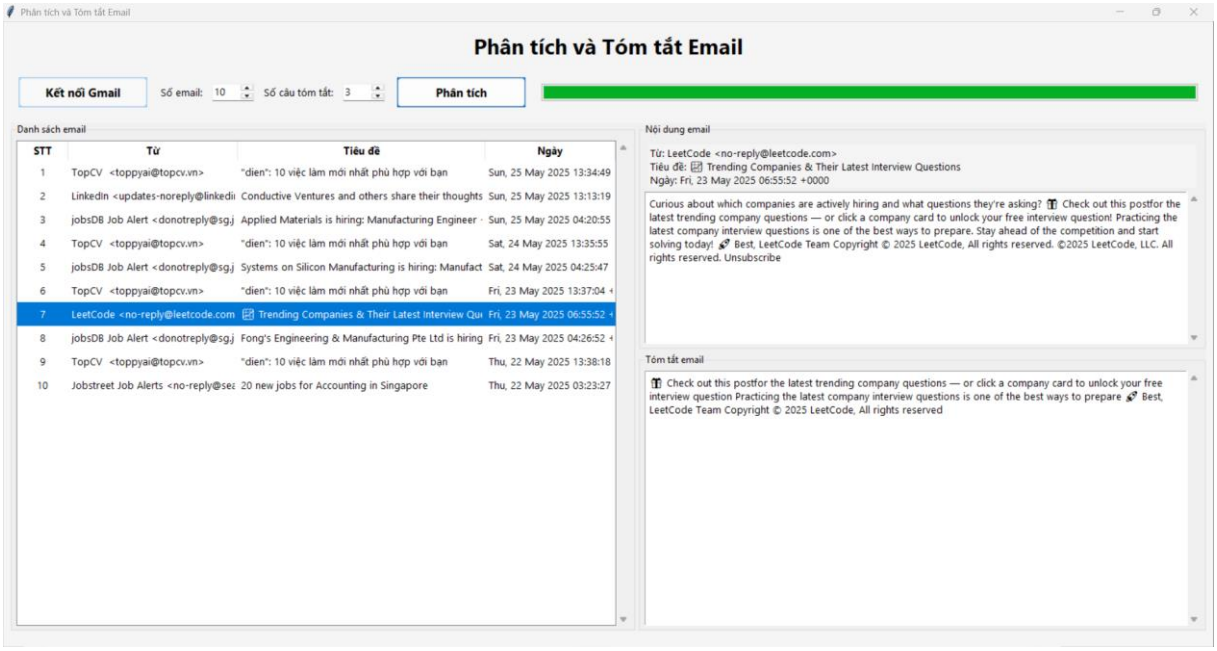
Hình 4. 1 Giao diện của ứng dụng Tóm tắt Email

Hình 4.1 thể hiện giao diện chính của ứng dụng "Tóm tắt Email". Giao diện được thiết kế trực quan với các khu vực chức năng được phân chia rõ ràng:

1. Phần trên cùng: Tiêu đề và thanh công cụ
2. Bên trái: Danh sách email
3. Bên phải trên: Nội dung email đầy đủ
4. Bên phải dưới: Bản tóm tắt email
5. Dưới cùng: Thanh trạng thái

Giao diện hỗ trợ thay đổi số lượng email cần phân tích (từ 1 đến 50) và số câu trong bản tóm tắt (từ 1 đến 10). Thanh tiến trình hiển thị quá trình phân tích email, giúp người dùng theo dõi trạng thái của tác vụ.

4.3 Kết quả tóm tắt



Hình 4. 2 Kết quả phân tích và tóm tắt email

Hình 4-2 minh họa kết quả tóm tắt của một email mẫu. Bản tóm tắt giữ lại các thông tin quan trọng từ nội dung gốc, giúp người đọc nắm bắt được nội dung chính mà không cần đọc toàn bộ email.

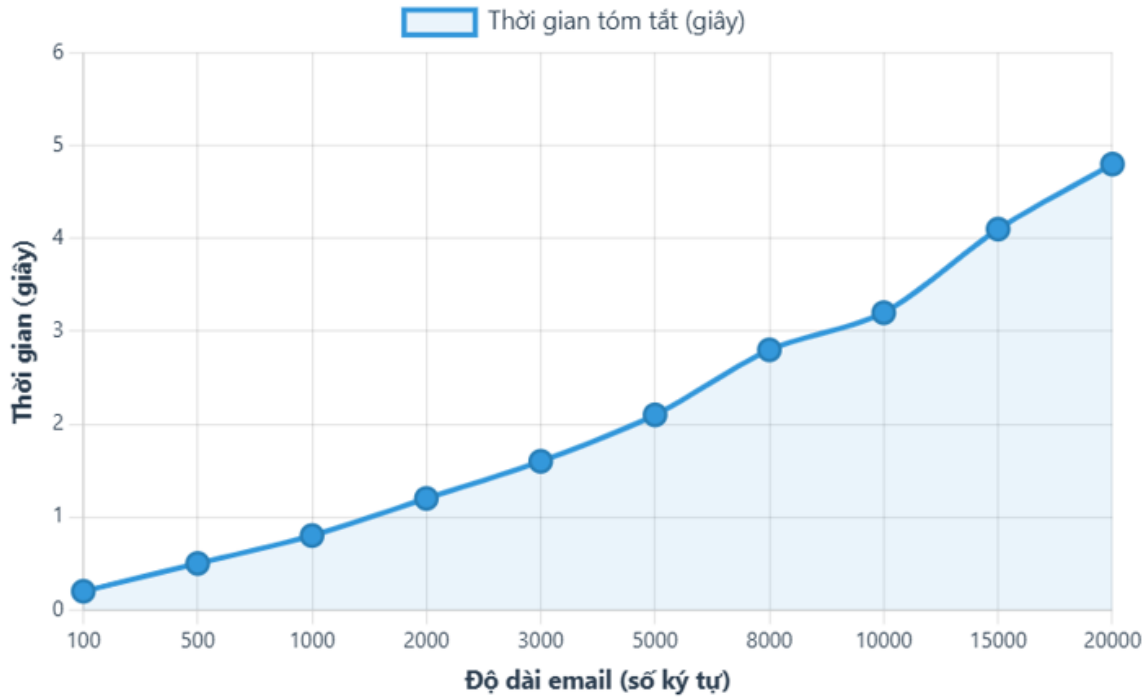
Bảng 4-2 trình bày kết quả đánh giá độ chính xác của thuật toán tóm tắt trên 50 email mẫu. Đánh giá được thực hiện dựa trên ý kiến của 5 người dùng thử nghiệm, mỗi người đánh giá mức độ chính xác của bản tóm tắt trên thang điểm từ 1 đến 5.

Loại email	**Số lượng**	**Điểm trung bình**	**Độ lệch chuẩn**
Email ngắn (<500 từ)	15	4.7	0.3
Email trung bình (500-1000 từ)	20	4.3	0.5
Email dài (>1000 từ)	15	3.9	0.7
Tổng cộng	50	4.3	0.6

Bảng 4. 2 Kết quả đánh giá độ chính xác của thuật toán tóm tắt

Kết quả cho thấy thuật toán tóm tắt hoạt động hiệu quả nhất với các email ngắn và trung bình, đạt điểm trung bình lần lượt là 4.7 và 4.3. Đối với email dài, chất lượng tóm tắt giảm nhẹ với điểm trung bình là 3.9, chủ yếu do khó khăn trong việc chọn lọc thông tin từ lượng nội dung lớn.

4.4 Đánh giá hiệu năng



Hình 4. 3 Kết quả phân tích và tóm tắt email

Bảng 4-3 trình bày thời gian phản hồi trung bình của ứng dụng cho các tác vụ chính. Thời gian được đo trên máy tính có cấu hình như trong Bảng 4-1.

Tác vụ	**Thời gian trung bình (giây)**
Kết nối Gmail	3.2
Truy xuất 10 email	4.8
Truy xuất 50 email	18.5
Hiển thị nội dung email	0.3
Tóm tắt email ngắn	0.5
Tóm tắt email trung bình	1.2
Tóm tắt email dài	2.8

Bảng 4. 3 Thời gian phản hồi của ứng dụng

Hình 4-3 minh họa mối quan hệ giữa thời gian tóm tắt và độ dài của email. Kết quả cho thấy thời gian tóm tắt tăng gần như tuyến tính với độ dài của email, nhưng vẫn duy trì dưới 3 giây cho hầu hết các trường hợp.

Trong quá trình thử nghiệm, ứng dụng xử lý thành công các email có độ dài lên đến 80,000 ký tự mà không gặp vấn đề về hiệu suất hoặc bộ nhớ. Việc sử dụng đa luồng giúp giao diện người dùng vẫn phản hồi mượt mà trong quá trình thực hiện các tác vụ nặng như truy xuất và phân tích email.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Dự án " Tóm tắt Email thông minh" đã được triển khai thành công với các kết quả đạt được:

1. **Xây dựng thành công ứng dụng desktop** kết nối và truy xuất email từ Gmail với giao diện trực quan, dễ sử dụng.
2. **Phát triển thuật toán tóm tắt văn bản tiếng Việt** hiệu quả, có khả năng xử lý các email có nội dung và độ dài khác nhau.
3. **Tối ưu hóa hiệu năng** của ứng dụng thông qua việc sử dụng đa luồng, cho phép giao diện người dùng luôn phản hồi trong quá trình thực hiện các tác vụ nặng.
4. **Hỗ trợ tiếng Việt** với thuật toán tóm tắt được điều chỉnh để phù hợp với đặc thù của ngôn ngữ này.

Qua việc thử nghiệm với người dùng thực tế, ứng dụng đã chứng minh được khả năng giúp người dùng tiết kiệm thời gian trong việc đọc và xử lý email, đặc biệt với những email có nội dung dài.

Ưu điểm:

- Giao diện trực quan, dễ sử dụng
- Thuật toán tóm tắt hiệu quả cho tiếng Việt
- Thời gian phản hồi nhanh
- Kết nối an toàn với Gmail thông qua OAuth 2.0

Khuyết điểm:

- Chất lượng tóm tắt giảm với email có nội dung phức tạp
- Chỉ hỗ trợ Gmail, chưa tích hợp với các dịch vụ email khác
- Chưa hỗ trợ tóm tắt nội dung đính kèm (như tài liệu PDF, Word)

5.2 Hướng phát triển

Dựa trên kết quả đạt được và những hạn chế hiện tại, các hướng phát triển tiềm năng cho ứng dụng bao gồm:

1. **Cải tiến thuật toán tóm tắt:**
    - Tích hợp mô hình học máy để nâng cao chất lượng tóm tắt
    - Phát triển phương pháp tóm tắt trừu tượng (abstractive summarization)
    - Huấn luyện mô hình chuyên biệt cho các loại email khác nhau (công việc, cá nhân, quảng cáo)
2. **Mở rộng tính năng:**
    - Hỗ trợ nhiều dịch vụ email khác (Outlook, Yahoo Mail, ProtonMail)
    - Phân loại email tự động
    - Tóm tắt tệp đính kèm (PDF, Word, PowerPoint)
    - Phát hiện và phân tích cảm xúc trong email
3. **Phát triển phiên bản đa nền tảng:**
    - Xây dựng phiên bản web
    - Phát triển ứng dụng di động cho iOS và Android
    - Tích hợp với các công cụ văn phòng và quản lý công việc
4. **Tối ưu hóa hiệu năng:**
    - Cải thiện hiệu suất với email dài và phức tạp
    - Triển khai lưu trữ cache để giảm thời gian truy xuất
    - Tối ưu hóa sử dụng bộ nhớ
5. **Hỗ trợ thêm ngôn ngữ:**
    - Mở rộng thuật toán tóm tắt cho các ngôn ngữ khác
    - Cung cấp tính năng dịch tự động giữa các ngôn ngữ

6. TÀI LIỆU THAM KHẢO

1. Mihalcea, R., & Tarau, P. (2004). "TextRank: Bringing Order into Texts." Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 404-411.
2. Luhn, H. P. (1958). "The Automatic Creation of Literature Abstracts." IBM Journal of Research and Development, 2(2), 159-165.
3. Google LLC. (2022). "Gmail API Documentation." https://developers.google.com/gmail/api/
4. Tkinter Documentation. (2022). "Tkinter 8.6 Reference: A GUI for Python." https://docs.python.org/3/library/tkinter.html
5. Vu, T., Nguyen, D. Q., Nguyen, D. Q., Dras, M., & Johnson, M. (2018). "VnCoreNLP: A Vietnamese Natural Language Processing Toolkit." Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, 56-60.
6. Undertheseа Team. (2022). "Undertheseа: Vietnamese NLP Toolkit." https://github.com/undertheseanlp/undertheseа
7. Bird, S., Klein, E., & Loper, E. (2009). "Natural Language Processing with Python." O'Reilly Media.
8. OAuth 2.0 Authorization Framework. (2012). RFC 6749. https://tools.ietf.org/html/rfc6749

7. PHỤ LỤC

Mã nguồn chính:

7.1 Lớp VietnameseTextSummarizer

```
class VietnameseTextSummarizer:

    def __init__(self):

        self.vietnamese_stop_words = {

            'bị', 'bởi', 'cả', 'các', 'cái', 'cần', 'càng', 'chỉ', 'chiếc',

            'cho', 'chứ', 'chưa', 'chuyện', 'có', 'có thể', 'cứ', 'của', 'cùng',

            'cũng', 'đã', 'đang', 'để', 'đến nỗi', 'đều', 'điều', 'do', 'đó',

            'được', 'dưới', 'gì', 'khi', 'không', 'là', 'lại', 'lên', 'lúc',

            'mà', 'mỗi', 'một cách', 'này', 'nên', 'nếu', 'ngay', 'nhiều',

            'như', 'nhưng', 'những', 'nơi', 'nữa', 'phải', 'qua', 'ra', 'rằng',

            'rất', 'rồi', 'sau', 'sẽ', 'so', 'sự', 'tại', 'theo', 'thì', 'trên',

            'trước', 'từ', 'từng', 'và', 'vẫn', 'vào', 'vậy', 'vì', 'việc',

            'với', 'vừa'

        }

        self.stop_words = self.vietnamese_stop_words.union(set(punctuation))


    def preprocess_text(self, text):

        text = re.sub(r'\s+', ' ', text)

        text = re.sub(r'http[s]?://\S+', '', text)
```

```
        text = re.sub(r'\S+@\S+', '', text)
        text = text.strip()
        return text

    def split_sentences(self, text):
        text = self.preprocess_text(text)
        sentence_endings = r'([.!?]|\.{3})+(?=\s|$)'
        sentences = re.split(sentence_endings, text)
        sentences = [s.strip() for s in sentences if s and s.strip()]
        return sentences

    def split_words(self, sentence):
        words = vi_tokenize(sentence.lower())
        return [w for w in words if w not in self.stop_words]

    def summarize(self, text, num_sentences=3):
        try:
            if not text or not text.strip():
                return "Không có nội dung để tóm tắt."

            sentences = self.split_sentences(text)
```

```
if len(sentences) <= num_sentences:
    return text

word_freq = {}
for sentence in sentences:
    for word in self.split_words(sentence):
        word_freq[word] = word_freq.get(word, 0) + 1

sentence_scores = {}
for sentence in sentences:
    if len(sentence.split()) < 4:
        continue
    for word in self.split_words(sentence):
        if word in word_freq:
            if sentence not in sentence_scores:
                sentence_scores[sentence] = word_freq[word]
            else:
                sentence_scores[sentence] += word_freq[word]

summary_sentences = nlargest(
    num_sentences, sentence_scores, key=sentence_scores.get)
```

```
            summary_sentences.sort(key=lambda x: sentences.index(x))


            return ' '.join(summary_sentences)
        except Exception as e:
            print(f"Lỗi khi tóm tắt: {str(e)}")
            return text
```

7.2 Phương thức connect của GmailAnalyzer

```
def connect(self):
    try:
        creds = None
        if os.path.exists('token.json'):
            creds = Credentials.from_authorized_user_file(
                'token.json', self.SCOPES)
        if not creds or not creds.valid:
            if creds and creds.expired and creds.refresh_token:
                creds.refresh(Request())
            else:
                flow = InstalledAppFlow.from_client_secrets_file(
                    'credentials.json', self.SCOPES)
                creds = flow.run_local_server(port=0)
            with open('token.json', 'w') as token:
```

```
            token.write(creds.to_json())


        self.service = build('gmail', 'v1', credentials=creds)
        return True
    except Exception as e:
        print(f"Lỗi kết nối: {str(e)}")
        return False
```