LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn các thầy, cô giáo trong Khoa Công nghệ thông tin, trường Đại học Điện Lực, đã tạo điều kiện cho em thực hiện đề tài này.

Để có thể hoàn thành báo cáo đề tài “Nhận dạng từ văn bản sang tiếng nói”, nhóm em xin gửi lời cảm ơn chân thành nhất tới cô*TS. Nguyễn Thị Thu Hà*, đã truyền đạt, giảng dạy cho chúng em những kiến thức, những kinh nghiệm qúy báu trong thời gian học tập và rèn luyện, tận tình hướng dẫn chúng em trong quá trình làm báo cáo này.

Nhóm em cũng gửi lời cảm ơn tới bạn bè đã đóng góp những ý kiến quý bàu để nhóm em có thể hoàn thành báo cáo tốt hơn.

Tuy nhiên, do thời gian và trình độ có hạn nên báo cáo này chắc chắn không tránh khỏi những thiếu sót, nhóm em rất mong được sự đóng góp ý kiến của các thầy và toàn thể các bạn. Nhóm em xin chân thành cảm ơn.

Một lần nữa, em xin chân thành cảm ơn và luôn mong nhận được sự đóng góp quý báu của tất cả mọi người.

*Hà Nội, Tháng 6 Năm 2016*

**Nhóm sinh viên thực hiện**

Ngô Xuân Bách

Trần Anh Đức

LỜI NÓI ĐẦU

Trên [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh), **tổng hợp giọng nói** là việc tạo ra giọng nói của người từ đầu vào là văn bản hay các mã hóa việc [phát âm](https://vi.wikipedia.org/wiki/Ph%C3%A1t_%C3%A2m). Hệ thống này còn được gọi là **văn bản-sang-tiếng nói** (*text-to-speech*, **TTS**); tuy rằng không phải hệ thống tổng hợp giọng nói nào cũng có đầu vào là văn bản (nhiều hệ thống thu nhận mã hóa cách phát âm, ví dụ mã [IPA](https://vi.wikipedia.org/wiki/B%E1%BA%A3ng_m%E1%BA%ABu_t%E1%BB%B1_phi%C3%AAn_%C3%A2m_qu%E1%BB%91c_t%E1%BA%BF), như đầu vào). Hệ thống thực hiện việc này còn gọi là **máy tổng hợp giọng nói** (*text to speech engine*), có thể là hệ thống [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) hoặc [phần cứng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_c%E1%BB%A9ng).

Tính năng **nhận dạng văn bản thành giọng nói** rất hữu ích trong nhiều trường hợp, đặc biệt đối với các phần mềm từ điển, giúp cho bạn có thể nghe phát âm của từ một cách đơn giản, ngoài ra một số phần mềm khác có thể đọc giúp bạn nội dung đang hiển thị trên màn hình, đọc tin nhắn hay số điện thoại đang gọi, hướng dẫn chỉ đường bằng giọng nói...

Ngoài ra, **nhận dạng văn bản thành giọng nói** còn được ứng dụng trong nhiều công việc khác như: hỗ trợ người khiếm thị có thể giao tiếp bình thường được với máy tính, bỏ qua mọi rào về tàn tật. Hoặc ứng dụng trong các máy trả lời tự động như máy **ATM**.

Càng ngày, công nghệ càng phát triển dẫn tới hệ quả là đòi hỏi của con người ngày càng cao, dẫn tới sự ra đời của hang loạt trợ lí ảo như **Siri** của **Apple** hay **Cortana** của **Microsoft**. Để có thể giao tiếp được với con người, các trình trợ lý ảo phải có khả năng “nói chuyện” với người dung, qua đó trao đổi trực tiếp những nhu cầu, giải đáp thông tin một cách nhanh chóng mà không cần sự trợ giúp của con người.

Bằng sự trợ giúp của **Máy Học**, mà chúng ta có thể tạo ra được những phần mềm thông minh, có khả năng **Nhận dạng văn bản thành giọng nói**, giúp giao tiếp với con người một cách chủ động và thân thiện hơn.

Đề tài của chúng em xin phép nghiên, phân tích và cài đặt chương trình **“Nhận dạng văn bản thành giọng nói”** trên môi trường cài đặt là Visual Studio 2013, mã nguồn viết trên C#, sử dụng bộ thư viện **Microsoft Speech API (SAPI)** và thư viện **NHMTTS - Chuyển văn bản thành tiếng nói tiếng Việt (Web API)**.

Nhận xét của giáo viên

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

Phân công công việc

|  |  |
| --- | --- |
| **Thành viên** | **Công việc** |
| Ngô Xuân Bách | …. |
| Trần Anh Đức | ….. |

Mục Lục

[Chương 1: Giới thiệu máy học 6](#_Toc453415965)

[1.1 Khái niệm máy học 6](#_Toc453415966)

[1.2 Ứng dụng học máy 6](#_Toc453415967)

[1.3 Các loại thuật toán học máy 6](#_Toc453415968)

[1.4 Các ví dụ học máy 6](#_Toc453415969)

[Chương 2: Nhận dạng từ văn bản sang tiếng nói 10](#_Toc453415970)

[2.1. Giới thiệu chung 10](#_Toc453415971)

[2.2. Lịch sử phát triển 11](#_Toc453415972)

[2.3. Công nghệ nhận dạng văn bản sang tiếng nói 12](#_Toc453415973)

[2.4. Kỹ thuật ngoại diện 12](#_Toc453415974)

[2.5. Các hệ điều hành có tổng hợp giọng nói 13](#_Toc453415975)

[2.6. Giới thiệu các thư viện sử dụng 15](#_Toc453415976)

[Chương 3: Phân tích chức năng 16](#_Toc453415977)

[Chương 4: Cài đặt chương trình 19](#_Toc453415978)

[Chương 5: Giao diện phần mềm 20](#_Toc453415979)

[Chương 6: Kiểm thử phần mềm 21](#_Toc453415980)

[Chương 7: Tài liệu hướng dẫn sử dụng 22](#_Toc453415981)

# Chương 1: Giới thiệu máy học

## 1.1 Khái niệm máy học

* Học máy là một phương pháp để tạo ra các chương trình máy tính bằng việc phân tích các tập dữ liệu.
* Học máy có liên quan lớn đến [thống kê](http://vi.wikipedia.org/wiki/Th%E1%BB%91ng_k%C3%AA), vì cả hai lĩnh vực đều nghiên cứu việc phân tích dữ liệu, nhưng khác với thống kê, học máy tập trung vào sự phức tạp của các giải thuật trong việc thực thi tính toán
* Máy học tập trung nghiên cứu và phát triển các vấn đề của trí tuệ nhân tạo, của các hệ thống máy thông minh dựa trên các cách thức thu thập tri thức, các cách thức biểu diễn tri thức, các cách thức xử lý tri thức.
* Máy học được ứng dụng trong một số các lĩnh vực như: máy truy tìm dữ liệu, chuẩn đoán trong y khoa, chuẩn đoán trong quá trình phân tích thị trường chứng khoán, phát hiện thẻ tín dụng giả, phân loại các chuỗi DNA, nhận dạng, xử lý tiếng nói, hình ảnh, chữ viết, quá trình dịch tự động giữa các ngôn ngữ, xây dựng các trò chơi, ứng dụng trong robot.

## 1.2 Ứng dụng học máy

Các ứng dụng Học máy bao gồm:

* Máy truy tìm dữ liệu.
* Chẩn đoán y khoa.
* Phát hiện thẻ tín dụng giả.
* Phân tích thị trường chứng khoán.
* Phân loại các chuỗi DNA, nhận dạng tiếng nói và chữ viết, dịch tự động, chơi trò chơi và cử động rô-bốt (robot locomotion).

## 1.3 Các loại thuật toán học máy

* Học có giám sát -- trong đó, thuật toán tạo ra một hàm ánh xạ dữ liệu vào tới kết quả mong muốn. Một phát biểu chuẩn về một việc học có giám sát là bài toán phân loại: chương trình cần học (cách xấp xỉ biểu hiện của) một hàm ánh xạ một vector tới một vài lớp bằng cách xem xét một số mẫu dữ\_liệu - kết\_quả của hàm đó.
* Học không giám sát -- mô hình hóa một tập dữ liệu, không có sẵn các ví dụ đã được gắn nhãn.
* Học nửa giám sát : kết hợp các ví dụ có gắn nhãn và không gắn nhãn để sinh một hàm hoặc một bộ phân loại thích hợp.
* Học tăng cường : trong đó, thuật toán học một chính sách hành động tùy theo các quan sát về thế giới. Mỗi hành động đều có tác động tới môi trường, và môi trường cung cấp thông tin phản hồi để hướng dẫn cho thuật toán của quá trình học.
* Chuyển đổi -- tương tự học có giám sát nhưng không xây dựng hàm một cách rõ ràng. Thay vì thế, cố gắng đoán kết quả mới dựa vào các dữ liệu huấn luyện, kết quả huấn luyện, và dữ liệu thử nghiệm có sẵn trong quá trình huấn luyện.
* Học cách học -- trong đó thuật toán học thiên kiến quy nạp của chính mình, dựa theo các kinh nghiệm đã gặp.

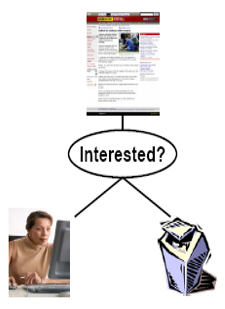
## 1.4 Các ví dụ học máy

* **Bài toán lọc các trang web theo sở thích của một người dung.**

T: Dự án (để lọc) xem những trang web nào mà một người dùng cụ thể thích đọc.

P: Tỷ lệ (%) các trang web được dự đoán đúng.

E: Một tập các trang web mà người dùng đã chỉ định là thích đọc và một tập các trang web mà anh ta chỉ định là không thích đọc.



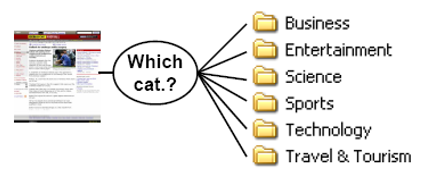
*Hình 1.1 Lọc các trang web theo sở thích*

* **Bài toán phân loại các trang web theo các chủ đề.**

T: Phân loại các trang web theo các chủ đề đã định trước.

P: Tỷ lệ (%) các trang web được phân loại chính xác.

E: Một tập các trang web, trong đó mỗi trang web gắn với một chủ đề.



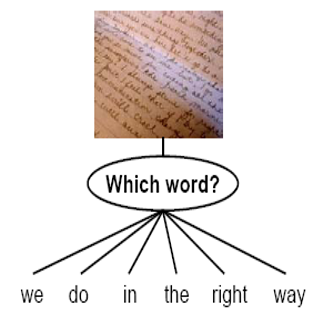
*Hình 1.2 Phân loại các trang web theo chủ đề*

* **Bài toán nhận dạng chữ viết tay.**

T: Nhận dạng và phân loại các từ trong các ảnh chữ viết tay.

P: Tỷ lệ (%) các từ được nhận dạng và phân loại đúng.

E: Một tập các ảnh chữ viết tay, trong đó mỗi ảnh được gắn với một định danh của một từ.



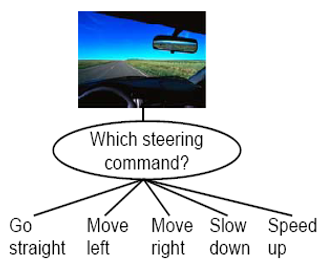
*Hình 1.3 Nhận dạng chữ viết tay*

* **Bài toán robot lái xe tự động.**

T: Robot (được trang bị các camera quan sát) lái xe tự động trên đường cao tốc.

P: Khoảng cách trung bình mà robot có thể lái xe tự động trước khi xảy ra lỗi.

E: Một tập các ví dụ được ghi lại khi quan sát một người lái xe trên đường cao ốc, trong đó mỗi ví dụ gồm một chuỗi các ảnh và các lệnh điều khiển.



*Hình 1.4 Robot lái xe tự động*

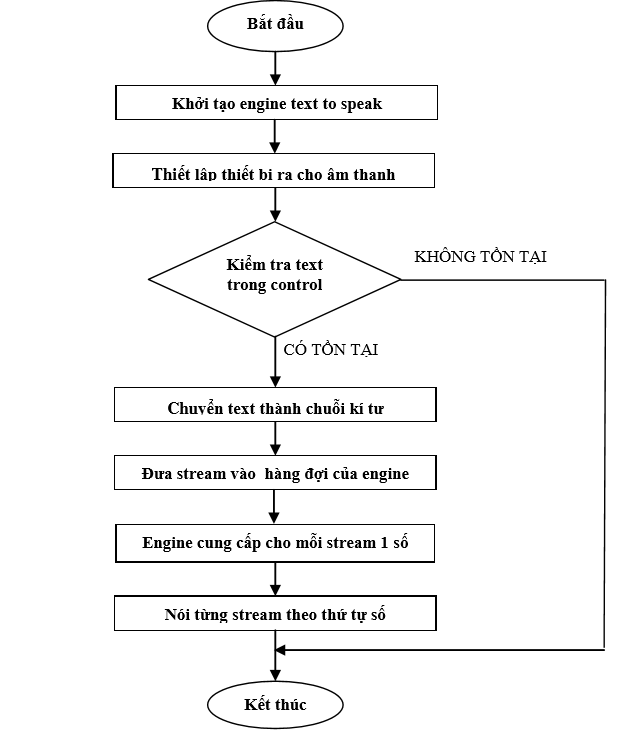
# Chương 2: Nhận dạng từ văn bản sang tiếng nói

## 2.1. Giới thiệu chung

Một máy **tổng hợp giọng nói** bao gồm hai phần: **ngoại diện** và **hậu trường**. Phần ngoại diện nhận [đầu vào](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A7u_v%C3%A0o&action=edit&redlink=1) ở dạng văn bản rồi cho [đầu ra](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A7u_ra) là [thể hiện biểu tượng ngôn ngữ](https://vi.wikipedia.org/w/index.php?title=Th%E1%BB%83_hi%E1%BB%87n_bi%E1%BB%83u_t%C6%B0%E1%BB%A3ng_ng%C3%B4n_ng%E1%BB%AF&action=edit&redlink=1) của văn bản (tức là một cách mã hóa cách phát âm văn bản). Phần hậu trường nhận lấy thể hiện biểu tượng ngôn ngữ như đầu vào và cho ra giọng nói tổng hợp ở [dạng sóng](https://vi.wikipedia.org/w/index.php?title=D%E1%BA%A1ng_s%C3%B3ng&action=edit&redlink=1) [âm thanh](https://vi.wikipedia.org/wiki/%C3%82m_thanh).

**Phần ngoại diện** có hai nhiệm vụ chính. Trước tiên, nó nhận văn bản và chuyển đổi các ký tự như các chữ số hay cách viết tắt thành dạng viết đầy đủ. Quá trình này gọi là *chuẩn hóa văn bản*, hay *tiền xử lý*. Sau đó nó cho ra [mã phát âm](https://vi.wikipedia.org/w/index.php?title=M%C3%A3_ph%C3%A1t_%C3%A2m&action=edit&redlink=1) ứng với từng từ, rồi phân chia và đánh dấu văn bản thành từng [đoạn văn](https://vi.wikipedia.org/w/index.php?title=%C4%90o%E1%BA%A1n_v%C4%83n&action=edit&redlink=1), [nhóm từ](https://vi.wikipedia.org/w/index.php?title=Nh%C3%B3m_t%E1%BB%AB&action=edit&redlink=1), [mệnh đề](https://vi.wikipedia.org/wiki/M%E1%BB%87nh_%C4%91%E1%BB%81), hay [câu văn](https://vi.wikipedia.org/w/index.php?title=C%C3%A2u_v%C4%83n&action=edit&redlink=1). Quá trình chuyển văn bản sang mã phát âm được gọi là *văn bản-sang-âm vị*. Kết hợp mã phát âm và thông tin đoạn văn tạo nên đầu ra cuối cùng *thể hiện biểu tượng ngôn ngữ*.

**Phần hậu trường**, nhận lấy thể hiện biểu tượng ngôn ngữ rồi chuyển nó thành âm thanh. Phần này thường được gọi là **máy tổng hợp**.



*Mô tả hoạt động của hệ thống nhận dạng văn bản thành âm thanh*

## 2.2. Lịch sử phát triển

Từ lâu trước khi kỹ thuật xử lý tín hiệu bằng thiết bị điện tử hiện đại ra đời, các nhà nghiên cứu giọng nói đã cố gắng xây dựng các máy móc bắt chước giọng nói của người. Các máy tổng hợp giọng nói bằng kỹ thuật điện tử, trong giai đoạn này, có giọng nói không tự nhiên và khó nghe. Tuy nhiên, chất lượng tổng hợp giọng nói ngày càng được cải tiến, cho đến ngày nay giọng phát âm của nhiều hệ thống có chất lượng tương đương giọng nói của người thật. Các hệ thống tổng hợp giọng nói đầu tiên được tạo ra vào những năm 1950 và hệ thống hoàn chỉnh đầu tiên ra đời vào năm 1968.

Công nghệ tổng hợp giọng nói đã tiến hóa nhanh kể từ đó. Hiện nay có hàng trăm hệ thống tổng hợp giọng nói, thương mại cũng như tự do.

## 2.3. Công nghệ nhận dạng văn bản sang tiếng nói

Hai tính chất quan trọng của chất lượng hệ thống tổng hợp giọng nói *là mức độ tự nhiên* và *mức độ dễ nghe*. Mức độ tự nhiên của giọng nói tổng hợp chỉ đến sự giống nhau giữa giọng tổng hợp và giọng nói tự nhiên của người thật. Mức độ dễ nghe chỉ đến việc câu phát âm có thể hiểu được dễ dàng không. Một máy tổng hợp giọng nói lý tưởng cần vừa tự nhiên vừa dễ nghe, và mục tiêu xây dựng máy tổng hợp giọng nói là làm gia tăng đến mức tối đa hai tính chất này. Một số hệ thống thiên về mức độ dễ nghe hơn, hoặc mức độ tự nhiên hơn; tùy thuộc vào mục đích mà công nghệ được lựa chọn. Có hai công nghệ chính được dùng là tổng hợp ghép nối và tổng hợp cộng hưởng tần số; ngoài ra cũng có một số công nghệ khác.

## 2.4. Kỹ thuật ngoại diện

**Chuẩn hóa văn bản**

Quá trình chuẩn hóa văn bản thường không đơn giản. Lý do là các văn bản thường chứa nhiều từ đồng tự, số và từ viết tắt đòi hỏi hiểu để diễn đạt lại trong văn bản đầy đủ.

Trong một số ngôn ngữ, các từ có thể được phát âm khác nhau tùy theo ngữ cảnh. Đa số hệ thống tổng hợp giọng nói không tạo ra thể hiện văn phạm cho văn bản, vì quá trình này hiện chưa có công nghệ đáng tin cậy. Thay vào đó, nhiều cách lần mò được dùng để phân biệt các cách phát âm, như tìm các từ kế cận hay dùng thống kê về tần số xuất hiện.

Việc chọn cách phát âm số cũng là một vấn đề. Lý do là cũng có nhiều cách phất âm số tùy theo văn cảnh. Như 1325 có thể đọc "một nghìn ba trăm hai mươi nhăm" nếu nó là một số tự nhiên, nhưng cũng có thể là "một ba hai năm" nếu nó là bốn số mật mã ngân khoản. Thường hệ thống tổng hợp giọng nói có thể đoán văn cảnh bằng việc quan sát các từ kế cận, các số hay dấu câu bên cạnh, hoặc dùng trường hợp mặc định khi không thể phân định.

Tương tự, các cách viết tắt cũng có thể mang nhiều nghĩa, tùy thuộc quy ước của người viết.

**Văn bản sang âm vị**

Các hệ thống tổng hợp giọng nói dùng hai cách cơ bản để xác định cách phát âm cho một từ, một quá trình còn được gọi là chuyển đổi văn bản-sang-âm vị hay tự vị-sang-âm vị, vì [âm vị](https://vi.wikipedia.org/wiki/%C3%82m_v%E1%BB%8B) là thuật ngữ dùng bởi các [nhà ngôn ngữ học](https://vi.wikipedia.org/wiki/Nh%C3%A0_ng%C3%B4n_ng%E1%BB%AF_h%E1%BB%8Dc) để mô tả các âm khác nhau trong ngôn ngữ.

Cách thứ nhất, và đơn giản nhất, là **dựa vào từ điển**, sử dụng một từ điển lớn chứa tất cả các từ của một ngôn ngữ và chứa cách phát âm đúng tương ứng cho từng từ, lưu trong máy tính. Việc xác định cách phát âm đúng cho một từ chỉ đơn giản là tra trong từ điển và thay đoạn văn bản bằng mã phát âm đã ghi trong từ điển.

Cách thứ hai là **dựa trên quy tắc**, sử dụng các quy tắc phát âm để tìm ra cách phát âm tương ứng cho mỗi từ phù hợp với quy tắc.

Mỗi cách đều có ưu điểm và nhược điểm. cách dựa trên từ điển nhanh và chính xác, nhưng sẽ không hoạt động nếu từ cần phát âm không có trong từ điển và lượng từ vựng cần lưu là lớn. Cách dùng quy tắc hoạt động với mọi văn bản (miễn là phù hợp với quy tắc) nhưng độ phức tạp của các quy tắc có thể tăng cao nếu ngôn ngữ có nhiều trường hợp bất quy tắc trong phát âm. Hầu hết các hệ thống tổng hợp giọng nói đều dùng kết hợp cả hai cách.

Một số ngôn ngữ, như [tiếng Tây Ban Nha](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_T%C3%A2y_Ban_Nha) hay [tiếng Việt](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Vi%E1%BB%87t), có hệ thống viết dựa trên cách phát âm một cách rất có quy tắc, và việc tiên đoán cách phát âm từ cách viết thường có tỷ lệ thành công cao. Các hệ thống tổng hợp giọng nói cho các ngôn ngữ này thường dùng chủ yếu cách dựa trên quy tắc, chỉ tra từ điển một vài từ đặc biệt như tên vay mượn từ nước ngoài.

Một số ngôn ngữ khác, như [tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh), có hệ thống phát âm rất bất quy tắc, thường cần hệ thống tổng hợp giọng nói dựa chủ yếu trên từ điển và dùng các quy tắc cho những từ không có trong từ điển.

## 2.5. Các hệ điều hành có tổng hợp giọng nói

**Mac OS và Mac OS X**

Hệ thống tổng hợp giọng nói đầu tiên được tích hợp vào trong một [hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) là [*Macintalk*](https://vi.wikipedia.org/w/index.php?title=Apple_PlainTalk&action=edit&redlink=1) trên máy tính [Macintosh](https://vi.wikipedia.org/wiki/Macintosh) của hãng [Apple Inc.](https://vi.wikipedia.org/wiki/Apple_Inc.) năm [1984](https://vi.wikipedia.org/wiki/1984). [Apple Inc.](https://vi.wikipedia.org/wiki/Apple_Inc.) là một trong những nhà sản xuất đầu tiên đưa hệ thống tổng hợp giọng nói vào các hệ điều hành thương mại. Trong những năm 1990, các giọng nói của Apple được tổng hợp từ các mẫu tự nhiên. Tuy nhiên gần đây, Apple đã thêm các mẫu giọng nói tổng hợp, là Vicki và Bruce - đặt tên theo giáo sư và nghiên cứu sinh tại khoa [ngôn ngữ học](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_h%E1%BB%8Dc) [UCLA](https://vi.wikipedia.org/wiki/UCLA), những người đã cung cấp các mô hình giọng nói này. Các phần mềm đầu tiên chỉ có ý định gây sự tò mò cho khách hàng và không được Apple hỗ trợ trực tiếp; tuy nhiên hệ thống tổng hợp giọng nói của máy tính [Macintosh](https://vi.wikipedia.org/wiki/Macintosh) đã tiến hóa thành một chương trình được hỗ trợ đầy đủ cho người khiếm thị.

**Microsoft Windows**

Các hệ điều hành [Windows](https://vi.wikipedia.org/wiki/Windows) hiện đại dùng các hệ thống tổng hợp giọng nói dựa trên [**SAPI4**](https://vi.wikipedia.org/w/index.php?title=Speech_Application_Programming_Interface&action=edit&redlink=1) và [**SAPI5**](https://vi.wikipedia.org/w/index.php?title=Speech_Application_Programming_Interface&action=edit&redlink=1), kèm theo [máy nhận dạng giọng nói](https://vi.wikipedia.org/w/index.php?title=M%C3%A1y_nh%E1%BA%ADn_d%E1%BA%A1ng_gi%E1%BB%8Dng_n%C3%B3i&action=edit&redlink=1). SAPI 4.0 có mặt trên các hệ điều hành như Windows 9x.

Nhiều phần mềm, như mIRC, dùng nhiều chức năng trong SAPI 4.0 hay SAPI 5.0. Windows XP có phần mềm Narrator. Hầu hết các phần mềm tương thích với Windows như Notepad, Office hay Adobe Acrobat có thể dùng các tính năng tổng hợp giọng nói; tùy theo lựa chọn trên trình đơn sau khi đã cài đặt. Chúng cung cấp hỗ trợ cho người khiếm thị.

Một ví dụ về việc SAPI 5 cho phép một phần mềm kết hợp công nghệ của Microsoft thành một màn hình nền có tính tương tác cao là Talking desktop. Phần mềm này kết hợp chức năng nhận dạng giọng nói với các phát âm của SAPI 5.

**Microsoft Speech Server** là một gói hoàn chỉnh để tổng hợp và nhận dạng giọng nói, và có thể ứng dụng cho các hệ thống liên lạc điện thoại có máy tính dùng Windows.

## 2.6. Giới thiệu các thư viện sử dụng

**Để hoàn thành đồ án này, chúng em có sử dụng các thư viện được cung cấp như:**

**Microsoft SAPI 5**

Việc ứng dụng công nghệ SAPI của Microsoft giúp tiết kiệm thời gian xây dựng engine nhận dạng tùy theo mục đích của mỗi nhà phát triển phần mềm. Đồng thời, windows là hệ điều hành đa người dùng nên chương trình có thể tích hợp trong các ứng dụng dễ dàng hơn.

Đây là một công nghệ được cung cấp miễn phí bởi Microsoft, điều này góp phần làm giảm đáng kể các chi phí về mặt bản quyền.

**NHMTTS** (tác giả Nguyễn Hữu Minh)

**Tính năng:**

* Tổng hợp tiếng nói tiếng Việt từ văn bản với các thông số **ngữ điệu gồm: trường độ, cường độ và khoảng ngừng; được xác định bởi mô hình mạng nơ-ron.**
* Hỗ trợ chuẩn giao tiếp **SAPI5**.
* **Cho phép đọc văn bản chứa đồng thời tiếng Việt và một ngôn ngữ khác.** Sử dụng mã Unicode.
* Cung cấp **2 giọng đọc nam và nữ**: "NHMTTS Voice (Male)" và "NHMTTS Voice (Female)".
* Cho phép kết hợp tùy chọn một giọng đọc khác trong hệ thống kết hợp với một giọng NHMTTS để có thể đọc văn bản chứa đồng thời hai ngôn ngữ.
* Cho phép điều chỉnh tốc độ đọc và âm lượng trong quá trình đọc.
* Thông báo vị trí văn bản đang đọc và thông tin khẩu hình tương ứng.
* Tự động phát hiện các từ sai vị trí dấu thanh.
* Kèm theo chương trình TTSApp minh họa các tính năng của bộ đọc.

**Giấy phép sử dụng: Sử dụng miễn phí, không giới hạn tính năng.**

# Chương 3: Phân tích chức năng



*Hình 3.1: Biểu đồ các chức năng chính của phần mềm*

**Chức năng soạn thảo văn bản**

Chức năng cho phép soạn thảo trực tiếp nội dung cần nhận dạng từ văn bản sang tiếng nói thông qua một **TextBox**. Ngoài ra người dùng còn có thể nhập vào nội dung cho text box thông qua menu **File > Nhập vào** hoặc xuất nội dung từ TextBox ra file text thông qua menu **File > Xuất ra**.

Các file được sử dụng trong phần mềm này cần được lưu trữ, mã hóa bằng bảng mã Unicode UTF-8.

**Chức năng nhận dạng văn bản sang tiếng nói**

Đây là chức năng chính, cũng xem như bộ não của toàn bộ phần mềm. Chức năng này bao gồm 4 chức năng nhỏ:

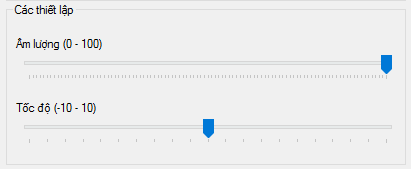
* *Chức năng phát âm* thực hiển đọc vào nội dung mà người dùng soạn thảo, tìm bộ phát âm phù hợp tiến hàng nhận dạng từ văn bản sang tiếng nói, sau đó thực hiện phát âm.
* *Chức năng tạm dừng, tiếp tục* cho phép người dùng tạm dừng quá trình phát âm lại, và có thể tiếp tục bất cứ lúc nào
* *Chức năng dừng* kết thúc quá trình phát âm
* *Chức năng xuất ra .WAV* cho phép người dùng lưu trữ kết quả phát âm lại dưới dạng file mềm.



*Hình 3.2: Biểu đồ biểu diễn quá trình hoạt động của chức năng nhận dang văn bản sang tiếng nói*

**Chức năng thiết lập chung**

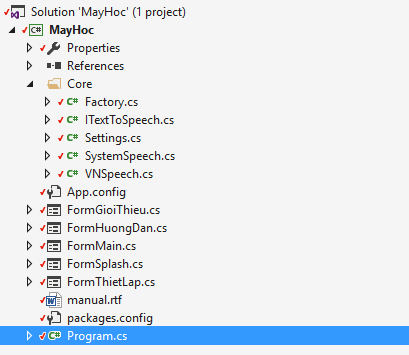
Khi sử dụng bộ phát âm SAPI để phát âm tiếng Anh, người dùng có thể thiết lập các thông số phát âm như **Âm lượng (Volume)** và **Tốc độ đọc (Rate)**. Để thực hiện thay đổi các thông số này, người dùng có thể sử dụng menu **Thiết lập > Thiết lập chung** để hiển thị Form thiết lập. Thông qua form này người dùng sẽ thực hiện các thay đổi mong muốn. Sau đó chọn Lưu để các thiết lập được thay đổi.



Hình 3.3: Các chức năng thiết lập

# Chương 4: Cài đặt chương trình

## 4.1. Thành phần của chương trình



Hình 4.1: Cấu trúc của project

**Các forms hiển thị**

* Form splash: hiển thị thông tin của phần mềm, tên đề tài, nhóm sinh viên thực hiện
* Form giới thiệu: hiển thị thông tin tác giả
* Form Main: form chính của chương trình, nơi người dùng giao tiếp với phần mềm
* Form hướng dẫn: hiển thị hướng dẫn sử dụng phần mềm
* Form thiết lập: cho phép người dùng thay đổi các thiết lập của phần mềm

**Các lớp hỗ trợ**

* Core.Factory: factory pattern, qua class này chúng ta có thể tạo ra các bộ phát âm tiếng Anh hoặc tiếng Việt dựa vào input của người dùng
* Core.ITextToSpeech: Interface của bộ phát âm, là nền tảng để các bộ phát âm có thể triển khai từ nó
* Core.Settings: class giúp lưu trữ và cung cấp các thiết lập của phần mềm
* Core.SystemSpeech: bộ phát âm tiếng Anh triển khai từ ITextToSpeech, sử dụng **SAPI 5** để phát âm tiếng Anh
* Core.VnSpeech: bộ phát âm tiếng Việt triển khai từ ITextToSpeech, sử dụng **NHMTTS** để phát âm tiếng Việt

**Các file khác**

* Manual.rtf: file hướng dẫn sử dụng, ở định dạng **Rich Text Format**

## 4.2. Mã nguồn chương trình

**FormMain.cs**

|  |
| --- |
| using System;  using System.IO;  using System.Windows.Forms;  using MayHoc.Core;  namespace MayHoc  {  public partial class FormMain : Form  {  private ITextToSpeech \_current;  public FormMain()  {  InitializeComponent();  }  private void FormMain\_Load(object sender, EventArgs e)  {  }  private void mnuExit\_Click(object sender, EventArgs e)  {  Close();  }  private void mnuIntroduce\_Click(object sender, EventArgs e)  {  new FormGioiThieu().ShowDialog();  }  private void mnuHuongDan\_Click(object sender, EventArgs e)  {  new FormHuongDan().ShowDialog();  }  private void mnuPreference\_Click(object sender, EventArgs e)  {  new FormThietLap().ShowDialog();  }  private void mnuSave\_Click(object sender, EventArgs e)  {  var dlg = new SaveFileDialog {Filter = "File text (\*.txt)|\*.txt"};  if (dlg.ShowDialog() == DialogResult.OK)  {  File.WriteAllText(dlg.FileName, txtContent.Text);  }  }  private void mnuOpen\_Click(object sender, EventArgs e)  {  var dlg = new OpenFileDialog { Filter = "File text (\*.txt)|\*.txt|Mọi định dạng (\*.\*)|\*.\*" };  if (dlg.ShowDialog() == DialogResult.OK)  {  txtContent.Text = File.ReadAllText(dlg.FileName);  }  }  private void btnPlay\_Click(object sender, EventArgs e)  {  if (\_current != null && !\_current.IsStopped())  {  MessageBox.Show("Trình phát giọng nói vẫn đang chạy!", "Lỗi", MessageBoxButtons.OK,  MessageBoxIcon.Error);  return;  }  if (\_current != null) \_current.Stop();  \_current = chkVietnamese.Checked ? new Factory().MakeVNSpeech(txtContent.Text) : new Factory().MakeLocal(txtContent.Text);  \_current.Speak();  }  private void btnPause\_Click(object sender, EventArgs e)  {  if (\_current == null) return;  \_current.PauseResume();  }  private void btnStop\_Click(object sender, EventArgs e)  {  if (\_current == null) return;  \_current.Stop();  \_current = null;  }  private void btnExportWav\_Click(object sender, EventArgs e)  {  var dlg = new SaveFileDialog {Filter = "Audio file (\*.wav)|\*.wav"};  if (dlg.ShowDialog() != DialogResult.OK) return;  // First, stop current playing voice  if (\_current == null) return;  \_current.Stop();  \_current = null;  var tmp = chkVietnamese.Checked ? new Factory().MakeVNSpeech(txtContent.Text) : new Factory().MakeLocal(txtContent.Text);  tmp.Export(dlg.FileName);  }  }  } |

**FormThietLap.cs**

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace MayHoc  {  public partial class FormThietLap : Form  {  public FormThietLap()  {  InitializeComponent();  }  private void btnSave\_Click(object sender, EventArgs e)  {  Properties.Settings.Default.Volume = trackVolume.Value;  Properties.Settings.Default.Rate = trackRate.Value;  Properties.Settings.Default.Save();  Close();  }  private void btnCancel\_Click(object sender, EventArgs e)  {  Close();  }  }  } |

**FormSplash.cs**

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace MayHoc  {  public partial class FormSplash : Form  {  private int \_counter;  public FormSplash()  {  InitializeComponent();  \_counter = 0;  }  private void timerFade\_Tick(object sender, EventArgs e)  {  \_counter++;  if (\_counter >= 40\*3)  {  timerFade.Enabled = false;  Hide();  var main = new FormMain();  main.FormClosed += (o, args) => Close();  main.Show();  }  else  {  if (\_counter < 40)  {  Opacity = \_counter\*0.025;  }  else if (\_counter > 80)  {  Opacity = (120 - \_counter)\*0.025;  }  else  {  Opacity = 1;  }  }  }  }  } |

**Core.Factory.cs**

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Speech;  using RestSharp;  namespace MayHoc.Core  {  public class Factory  {  public ITextToSpeech MakeLocal(string text)  {  return new SystemSpeech(text);  }  public ITextToSpeech MakeVNSpeech(string text)  {  return new VNSpeech(text);  }  }  } |

**Core.ItextToSpeech.cs**

|  |
| --- |
| namespace MayHoc.Core  {  public interface ITextToSpeech  {  void Speak();  void Export(string file);  void PauseResume();  void Stop();  bool IsStopped();  }  } |

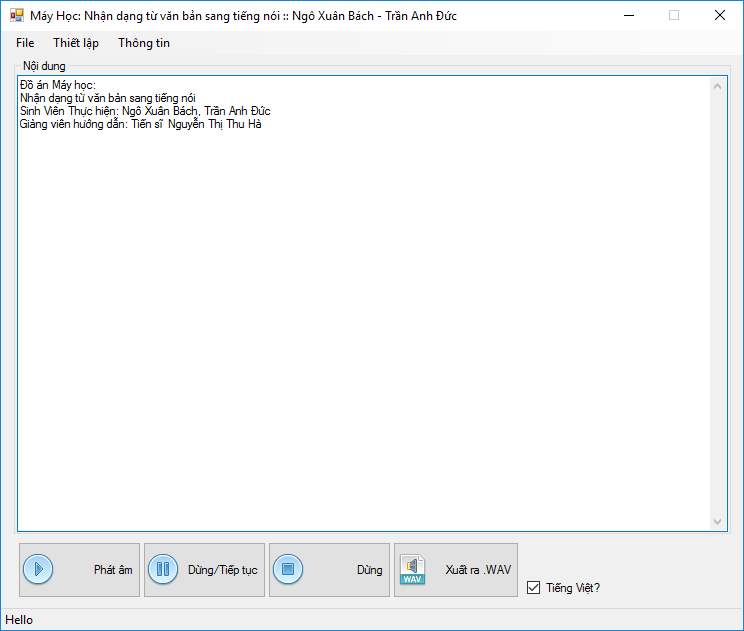
**Core.SystemSpeech.cs**

|  |
| --- |
| using System;  using System.Speech;  using System.Speech.Synthesis;  namespace MayHoc.Core  {  public class SystemSpeech : ITextToSpeech  {  private readonly string \_text;  private SpeechSynthesizer \_speaker;  public SystemSpeech(string text)  {  \_text = text;  }  public void Speak()  {  if (\_speaker != null) return;  \_speaker = new SpeechSynthesizer();  \_speaker.SpeakAsync(\_text);  \_speaker.Volume = Properties.Settings.Default.Volume;  \_speaker.Rate = Properties.Settings.Default.Rate;  }  public void Export(string file)  {  var tmp = new SpeechSynthesizer();  tmp.SetOutputToWaveFile(file);  tmp.Speak(\_text);  tmp.Dispose();  }  public void PauseResume()  {  if (\_speaker == null) return;  if (\_speaker.State == SynthesizerState.Speaking)  \_speaker.Pause();  else  \_speaker.Resume();  }  public void Stop()  {  if (\_speaker == null) return;  \_speaker.Dispose();  \_speaker = null;  }  public bool IsStopped()  {  return \_speaker.State == SynthesizerState.Ready;  }  }  } |

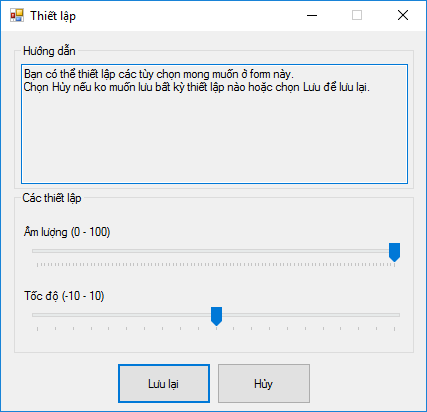
**Core.VNSpeech.cs**

|  |
| --- |
| using System;  using System.IO;  using RestSharp;  using NAudio.Wave;  namespace MayHoc.Core  {  public class VNSpeech : ITextToSpeech, IDisposable  {  private readonly byte[] \_data;  private readonly IWavePlayer waveOutDevice = new WaveOut();  private AudioFileReader audioFileReader;  public VNSpeech(string text)  {  var client = new RestClient("http://cloudtalk.vn/");  var req = new RestRequest("/tts", Method.POST);  req.AddParameter("text", text);  req.AddParameter("style", "poem");  req.AddParameter("uid", "reserved");  var id = client.Execute(req).Content;  req = new RestRequest("ttsoutput?id=" + id);  \_data = client.Execute(req).RawBytes;  File.WriteAllBytes("tmp.mp3", \_data);  }  public void Speak(){  audioFileReader = new AudioFileReader("tmp.mp3");  waveOutDevice.Init(audioFileReader);  waveOutDevice.Play();  }  public void Export(string file)  {  File.WriteAllBytes(file, \_data);  }  public void PauseResume()  {  if (waveOutDevice.PlaybackState == PlaybackState.Paused)  waveOutDevice.Play();  else  waveOutDevice.Pause();  }  public void Stop()  {  waveOutDevice.Stop();  Dispose();  }  public bool IsStopped()  {  return waveOutDevice.PlaybackState == PlaybackState.Stopped;  }  public void Dispose()  {  audioFileReader.Dispose();  waveOutDevice.Dispose();  }  }  } |

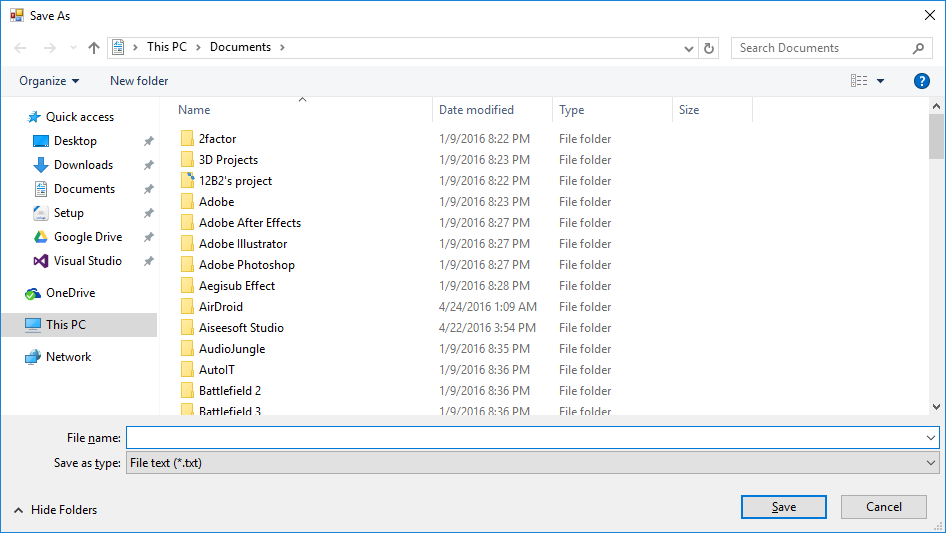
# Chương 5: Giao diện phần mềm



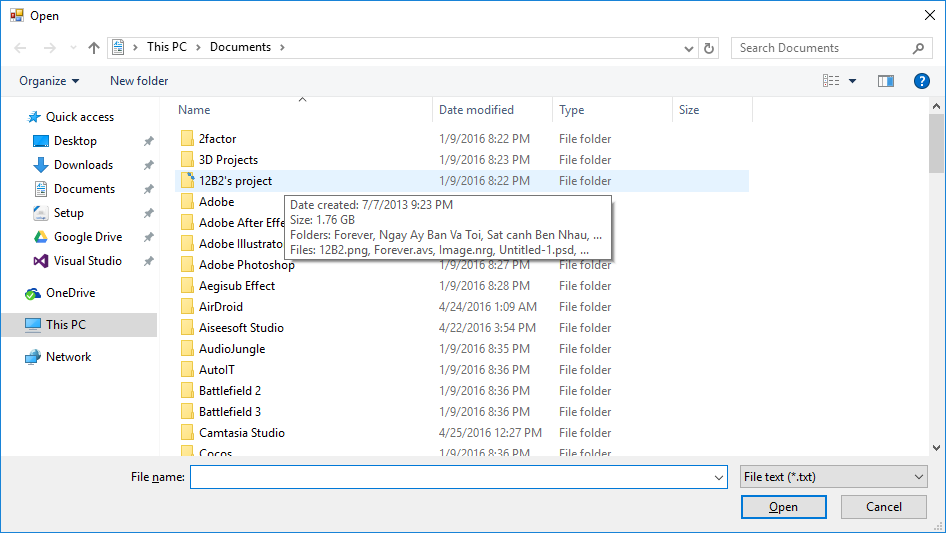
*Hình 5.1: Giao diện chính của phần mềm*



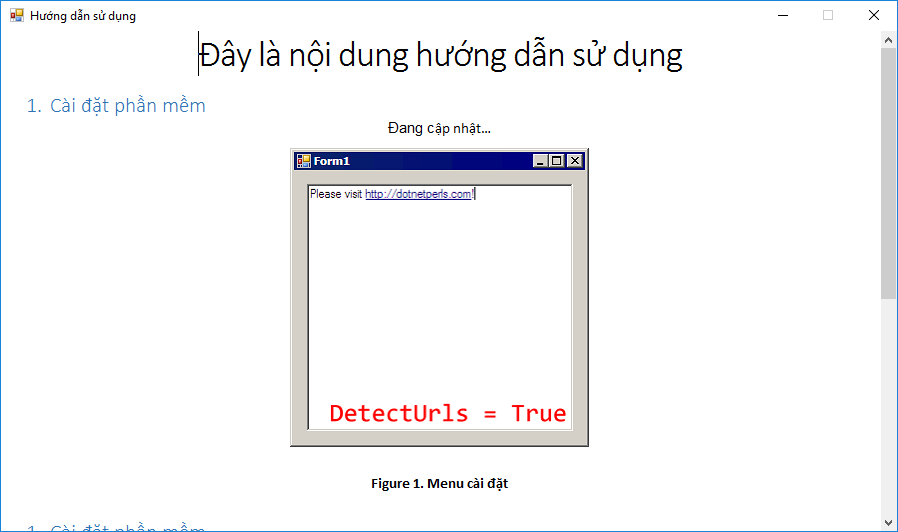
*Hình 5.2: Giao diện cửa sổ thiết lập chung*



*Hình 5.3: Giao diện của sổ lưu file*



*Hình 5.4: Giao diện cửa sổ đọc vào file*



*Hình 5.5: Giao diện cửa sổ hướng dẫn sử dụng*

# Chương 6: Tài liệu hướng dẫn sử dụng

Kết luận