LỜI CẢM ƠN

Em xin chân thành cảm ơn Bộ môn Công nghệ Thông tin - Trường Đại học Dân lập Hải Phòng đã tạo điều kiện cho em thực hiện đề tài tốt nghiệp này.

Em xin chân thành cảm ơn thầy Vũ Mạnh Khánh đã tận tình hướng dẫn, chỉ bảo cho em trong suốt thời gian thực hiện đề tài.

Em cũng xin chân thành cảm ơn quý Thầy Cô trong Bộ môn đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết trong suốt quá trình học tập tại trường. Em xin gửi lòng biết ơn sâu sắc đến bố, mẹ, gia đình, các bạn bè đã ủng hộ, giúp đỡ động viên em trong suốt quá trình học cũng như thời gian làm tốt nghiệp vừa qua.

Mặc dù đã rất cố gắng hoàn thành đồ án với tất cả nỗ lực của bản thân, nhưng do kiến thức bản thân còn có hạn nên chắc chắn đồ án không tránh khỏi những sai sót và hạn chế, kính mong sự thông cảm, chỉ bảo của quý Thầy Cô và các bạn.

Sinh viên

Nguyễn Việt Dũng

MỤC LỤC

LỜI NÓI ĐẦU

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ - công cụ hoàn hảo nhất của tư duy và giao tiếp.

Xử lý ngôn ngữ chính là xử lý thông tin khi đầu vào là "dữ liệu ngôn ngữ" (dữ liệu cần biến đổi), tức dữ liệu "văn bản" hay "tiếng nói". Các dữ liệu liên quan đến ngôn ngữ viết (văn bản) và nói (tiếng nói) đang dần trở nên kiểu dữ liệu chính con người có và lưu trữ dưới dạng điện tử. Đặc điểm chính của các kiểu dữ liệu này là không có cấu trúc hoặc nửa cấu trúc và chúng không thể lưu trữ trong các khuôn dạng cố định như các bảng biểu.

Xử lý ngôn ngữ tự nhiên là một lĩnh vực nghiên cứu nhằm giúp cho các hệ thống máy tính hiểu và xử lý được ngôn ngữ con người. Dịch máy là một trong những ứng dụng chính của xử lý ngôn ngữ tự nhiên. Để việc dịch được chính xác đòi hỏi phải phân tích chính xác cấu trúc ngữ pháp của câu đầu vào. Qua đó có thể sửa được lỗi của câu.

Chương 1: GIỚI THIỆU VỀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

1.1 Tổng quan

Xử lý ngôn ngữ chính là xử lý thông tin khi đầu vào là "dữ liệu ngôn ngữ" (dữ liệu cần biến đổi), tức dữ liệu "văn bản" hay "tiếng nói". Các dữ liệu liên quan đến ngôn ngữ viết (văn bản) và nói (tiếng nói) đang dần trở nên kiểu dữ liệu chính con người có và lưu trữ dưới dạng điện tử. Đặc điểm chính của các kiểu dữ liệu này là không có cấu trúc hoặc nửa cấu trúc và chúng không thể lưu trữ trong các khuôn dạng cố định như các bảng biểu. Theo đánh giá của công ty Oracle, hiện có đến 80% dữ liệu không cấu trúc trong lượng dữ liệu của loài người đang có [Oracle Text]. Với sự ra đời và phổ biến của Internet, của sách báo điện tử, của máy tính cá nhân, của viễn thông, của thiết bị âm thanh,… người người ai cũng có thể tạo ra dữ liệu văn bản hay tiếng nói. Vấn đề là làm sao ta có thể xử lý chúng, tức chuyển chúng từ các dạng ta chưa hiểu được thành các dạng ta có thể hiểu và giải thích được, tức là ta có thể tìm ra thông tin, tri thức hữu ích cho mình.

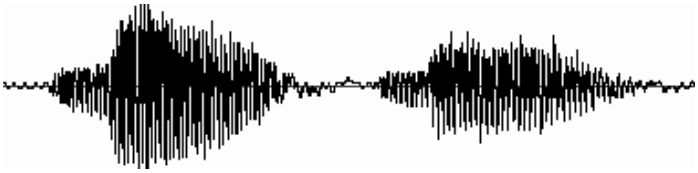
Giả sử chúng ta có các câu sau trong các tiếng nước ngoài:

- "We meet here today to talk about Vietnamese language and speech processing."

- "Aujourd'hui nous nous réunissons ici pour discuter le traitement de langue et de parole vietnamienne."

- "Мы встрачаемся здесь сегодня, чтобы говорить о вьетнамском языке и обработке речи."

Nếu có ai đó dịch, hoặc có một chương trình máy tính dịch (biến đổi) chúng ra tiếng Việt, ta sẽ hiểu nghĩa các câu trên đều là: "Hôm nay chúng ta gặp nhau ở đây để bàn về xử lý ngôn ngữ và tiếng nói tiếng Việt.". Nếu các câu này được lưu trữ như các tệp tiếng Anh, Pháp, Nga và Việt như ta nhìn thấy ở trên, ta có các dữ liệu "văn bản". Nếu ai đó đọc các câu này, ghi âm lại, ta có thể chuyển chúng vào máy tính dưới dạng các tệp các tín hiệu (signal) "tiếng nói". Tín hiệu sóng âm của hai âm tiết tiếng Việt có thể nhìn thấy như sau:



Hình 1.1 : Tín hiệu sóng âm của hai âm tiêt Tiếng Việt

Tuy nhiên, một văn bản thật sự (một bài báo khoa học chẳng hạn) có thể có đến hàng nghìn câu, và ta không phải có một mà hàng triệu văn bản. Web là một nguồn dữ liệu văn bản khổng lồ, và cùng với các thư viện điện tử − khi trong một tương lai gần các sách báo xưa nay và các nguồn âm thanh được chuyển hết vào máy tính (chẳng hạn bằng các chương trình nhận dạng chữ, thu nhập âm thanh, hoặc gõ thẳng vào máy) − sẽ sớm chứa hầu như toàn bộ kiến thức của nhân loại. Vấn đề là làm sao "xử lý" (chuyển đổi) được khối dữ liệu văn bản và tiếng nói khổng lồ này qua dạng khác để mỗi người có được thông tin và tri thức cần thiết từ chúng.

Xử lý ngôn ngữ tự nhiên đã được ứng dụng trong thực tế để giải quyết các bài toán như : nhận dạng chữ viết, nhận dạng tiếng nói, tổng hợp tiếng nói, dịch tự động, tìm kiếm thông tin, tóm tắt văn bản, khai phá dữ liệu và phát hiện tri thức.

1.2. Cơ sở khoa học

1.2.1. Một số khái niệm cơ bản

1.2.1.1. Ngôn ngữ tự nhiên

Ngôn ngữ là hệ thống để giao thiệp hay suy luận dùng một cách biểu diễn phép ẩn dụ và một loại ngữ pháp theo logic, mỗi cái đó bao hàm một tiêu chuẩn hay sự thật thuộc lịch sử và siêu việt. Nhiều ngôn ngữ sử dụng điệu bộ, âm thanh, ký hiệu, hay chữ viết, và cố gắng truyền khái niệm, ý nghĩa, và ý nghĩ, nhưng mà nhiều khi những khía cạnh này nằm sát quá, cho nên khó phân biệt nó.

1.2.1.2. Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ - công cụ hoàn hảo nhất của tư duy và giao tiếp.

1.2.1.3. Trí tuệ nhân tạo

Trí tuệ nhân tạo hay trí thông minh nhân tạo (tiếng Anh: artificial intelligence hay machine intelligence, thường được viết tắt là AI) là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào. Thuật ngữ này thường dùng để nói đến các máy tính có mục đích không nhất định và ngành khoa học nghiên cứu về các lý thuyết và ứng dụng của trí tuệ nhân tạo.

1.2.1.4. Nhập nhằng

Nhập nhằng trong ngôn ngữ học là hiện tượng thường gặp, trong giao tiếp hàng ngày con người ít để ý đến nó bởi vì họ xử lý tốt hiện tượng này. Nhưng trong các ứng dụng liên quan đến xử lý ngôn ngữ tự nhiên khi phải thao tác với ý nghĩa từ vựng mà điển hình là dịch tự động nhập nhằng trở thành vấn đề nghiêm trọng . Ví dụ trong một câu cần dịch có xuất hiện từ "đường" như trong câu "ra chợ mua cho mẹ ít đường" vấn đề nảy sinh là cần dịch từ này là road hay sugar, con người xác định chúng khá dễ dàng căn cứ vào văn cảnh và các dấu hiệu nhận biết khác nhưng với máy thì không. Một số hiện tượng nhập nhằng: Nhập nhằng ranh giới từ, Nhập nhằng từ đa nghĩa, Nhập nhằng từ đồng âm (đồng tự), Nhập nhằng từ loại.

1.2.1.5. Dịch máy

Dịch máy là một trong những ứng dụng chính của xử lý ngôn ngữ tự nhiên, dùng máy tính để dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác. Mặc dù dịch máy đã được nghiên cứu và phát triển hơn 50 năm qua, xong vẫn tồn tại nhiều vấn đề cần nghiên cứu. Ở Việt Nam, dịch máy đã được nghiên cứu hơn 20 năm, nhưng các sản phẩm dịch máy hiện tại cho chất lượng dịch còn nhiều hạn chế. Hiện nay, dịch máy được phân chia thành một số phương pháp như: dịch máy trên cơ sở luật, dịch máy thống kê và dịch máy trên cớ sở ví dụ.

1.2.2. Xác xuất (Probability)

1.2.2.1. Thực nghiêm và không gian mẫu

Không gian mẫu (sự kiện cơ sở): Ω.Tung từng đồng xu: Ω = {head,tail}

Bầu cử: Ω = {yes/no}.Tung xúc xắc Ω = {1,...,6} . Xổ số (|Ω | ≈ 10^7 .. 10^{12}).Số lượng tai nạn giao thông/năm (Ω = N) . Lỗi chính tả (Ω = Z*), Z là 1 bảng chữ cái, Z* là tập hợp các chuỗi trong bảng chữ cái (|Ω | ≈kích thước vốn từ vựng)

1.2.2.2. Events (sự kiện)

Sự kiện A là một tập các mẫu $A \subset \Omega$, và tập tất cả A là 2^{Ω} .Ω là sự kiện chắc chắn , Ø là sự kiện không xảy ra .Ví dụ : Tung đồng xu 3 lần

Ω = {HHH, HHT, HTH, HTT, THH, THT, TTH, TTT}.Tính các trường hợp có đúng 2 lần xuất hiện Tail . A = {HTT, THT, TTH} . Tất cả Head : A = {HHH}

1.2.2.3. Xác suất (probability)

Thực hiện một thực nghiệm (experiment) nhiều lần: có bao nhiêu lần sự kiện A xảy ra ("count" c1). Mỗi lần thực nghiệm này gọi là dãy (bộ) . Thực hiện các dãy này nhiều lần, ghi nhớ lại con số ci . Nếu thực hiện thật sự thực nghiệm nhiều lần, tỉ số ci/Ti (Ti là tổng số lần thực nghiệm trong dãy thứ i) dần tới một hằng số chưa biết . Gọi giá trị này Xác xuất của A . Kí hiệu: p(A)

1.2.2.4. Ước lượng Xác suất

Cách tính như sau:Từ một dãy thực nghiệm :p(A) = c1/T1.

Nếu thực hiện được nhiều dãy thực nghiệm: tính trung bình cộng của ci/Ti

1.2.2.5. Kỳ vọng (expectation) và Phương sai (variance)

Kỳ vọng: tổng trọng số của giá trị của X, hay là giá trị trung bình của biến ngẫu nhiên

Phương sai:là trung bình bình phương của độ lệch (độ lệch của biến X so với trung bình của nó)

$$E(X) = \sum_x xp(x)$$

$$Var(X) = \sum_x p(x)(x - E(x))^2$$

1.2.3.Lý thuyết thông tin(Information Theory)

1.2.3.1 Khái niệm

Lý thuyết thông tin nghiên cứu về: Áp dụng các công cụ toán học trong việc lượng hóa data cho mục đích lưu trữ và truyền dữ liệu. Độ đo thông tin là Entropy, là số lượng bít trung bình cần thiết để cho việc lưu trữ hay truyền dữ liệu.Đóng vai trò quan trọng trong xử lý thông tin bằng các phương pháp thống kê, đặc biệt trong NLP

1.2.3.2 Entropy

Entropy là một độ đo thông tin . Entropy ~ hỗn độn, mờ, trái nghĩa với order, .. Đo độ không chắc chắn : Entropy thấp -> Đo độ không chắc chắn thấp ; Entropy cao -> Đo độ không chắc chắn cao . Trong vật lý : Entropy giảm khi năng lượng được sử dụng . Ký hiệu p(x) là một phân bố của một biến ngẫu nhiên X . Ω là không gian mẫu của X . Entropy được tính như sau: H(X) = - ∑ x ∈Ω p(x) log2p(x) . Đơn vị: bits (log10: nats) . Kí hiệu: H(X) = Hp(X) = H(p)

1.2.3.3 Perplexity - Cross Entropy

1. Entropy liên quan thế nào đến hiểu ngôn ngữ?

Liên quan đến sự ko chính xác: một vấn đề càng có nhiều thông tin thì Entropy càng thấp.Có nhiều mô hình -> entropy đo chất lượng của các mô hình?

Ví dụ: mô hình mã hóa ký tự với trung bình số bít sử dụng trên mỗi ký tự là 2.5 Đây là mô hình ngôn ngữ 0-gram, nếu đặt trong sự liên kết của các âm tiết thì chúng ta có thể sinh được mô hình tốt hơn, chẳng hạn cho entropy 1.22 bít trên một ký tự

2. Perplexity

Entropy của một phân bố p(X) là :Hp(X)Thì giá trị 2H được gọi là perplexity perplexity là số lượng mẫu trung bình mà một biến phải lựa chọn.Perlexity càng bé (tức là entropy càng bé) thì mô hình càng tốt <=> số bít dùng để mã hóa thông tin càng bé.

Ví dụ : Cho 8 con ngựa với xác suất lựa chọn như sau:

Ngựa 1: 1/2 ngựa 2: 1/4 ngựa 3: 1/8 ngựa 4: 1/16

Ngựa 5: 1/64 ngựa 2: 1/64 ngựa 3: 1/64 ngựa 4: 1/64

3. Entropy rate

Tính entropy của một dãy các từ trong một ngôn ngữ L

H(w1,...,wn) = - ∑W∈L p(W1n)log(W1n)

Entropy rate được coi như per-word entropy.Coi một ngôn ngữ như một quá trình ngẫu nhiên sản xuất một dãy các từ. Cần quan tâm đến một dãy vô hạn từ. Entropy rate H(L) được định nghĩa như sau:

$$H(L) = \lim_{n->\infty} \frac{1}{n} H(w_1,..., w_n) = \lim_{n->\infty} -\frac{1}{n} \sum_{\in L} p(w_1,..., w_n) \log p(w_1,..., w_n)$$

4 . Cross Entropy

Cross entropy được sử dụng khi chúng ta không biết phân bố thật p

Cross-entropy của phân bố m của phân bố thật p được định nghĩa:

$$H(p,m) = \lim_{n->\infty} -\frac{1}{n} \sum_{\in L} p(w_1,...,w_n) \log m(w_1,...,w_n) = \lim_{n->\infty} -\frac{1}{n} \log m(w_1,...,w_n)$$

(theo lý thuyết Shannon-McMillan-Breiman)

5. Cross entropy để so sánh các mô hình : H(p) ≤ H(p,m)

Cross entropy H(p,m) là cận trên của entropy H(p)

Mô hình m càng chính xác thì cross entropy H(p,m) càng gần với entropy H(p)

Độ khác nhau H(p,m) và H(p) đo độ chính xác của mô hình m

6. Các công thức Cross Entropy

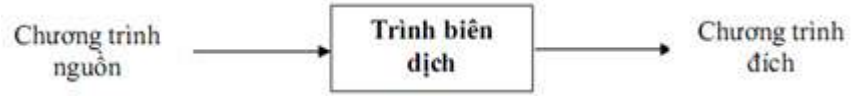
Cross entropy giữa biến X với phân bố xác suất đúng p(x) và một phân bố m được tính như sau:

$$H(X,m) = H(X) + D(p \parallel m) = -\sum_{x} p(x) \log m(x)$$

Chú ý:D(p||q) = ∑x p(x) log2 (p(x)/q(x))

1.3. Qui trình xử lý ngôn ngữ tự nhiên

Để máy tính có thể hiểu và thực thi một chương trình được viết bằng ngôn ngữ cấp cao, ta cần phải có một trình biên dịch thực hiện việc chuyển đổi chương trình đó sang chương trình ở dạng ngôn ngữ đích. Chương này trình bày một cách tổng quan về cấu trúc của một trình biên dịch và mối liên hệ giữa nó với các thành phần khác - "họ hàng" của nó - như bộ tiền xử lý, bộ tải và soạn thảo liên kết,v.v. Cấu trúc của trình biên dịch được mô tả trong chương là một cấu trúc mức quan niệm bao gồm các giai đoạn: Phân tích từ vựng, Phân tích cú pháp, Phân tích ngữ nghĩa, Sinh mã trung gian, Tối ưu mã và Sinh mã đích. Nói một cách đơn giản, trình biên dịch là một chương trình làm nhiệm vụ đọc một chương trình được viết bằng một ngôn ngữ - ngôn ngữ nguồn (source language) - rồi dịch nó thành một chương trình tương đương ở một ngôn ngữ khác - ngôn ngữ đích (target languague). Một phần quan trọng trong quá trình dịch là ghi nhận lại các lỗi có trong chương trình nguồn để thông báo lại cho người viết chương trình.



Hình: Một trình biên dịch

1.3.1. Phân tích từ vựng (Lexical Analysis)

Trong một trình biên dịch, giai đoạn phân tích từ vựng sẽ đọc chương trình nguồn từ trái sang phải (quét nguyên liệu - scanning) để tách ra thành các thẻ từ (token).

Ví dụ 1.2: Quá trình phân tích từ vựng cho câu lệnh gán position := initial + rate * 60 sẽ tách thành các token như sau:

1. Danh biểu position

2. Ký hiệu phép gán :=

3. Danh biểu initial

4. Ký hiệu phép cộng (+)

5. Danh biểu rate

6. Ký hiệu phép nhân (*)

7. Số 60

Trong quá trình phân tích từ vựng các khoảng trắng (blank) sẽ bị bỏ qua.

1.3.2. Phân tích cú pháp (Syntax Analysis)

Giai đoạn phân tích cú pháp thực hiện công việc nhóm các thẻ từ của chương trình nguồn thành các ngữ đoạn văn phạm (grammatical phrase), mà sau đó sẽ được trình biên dịch tổng hợp ra thành phẩm. Thông thường, các ngữ đoạn văn phạm này được biểu diễn bằng dạng cây phân tích cú pháp (parse tree) với :

- Ngôn ngữ được đặc tả bởi các luật sinh.

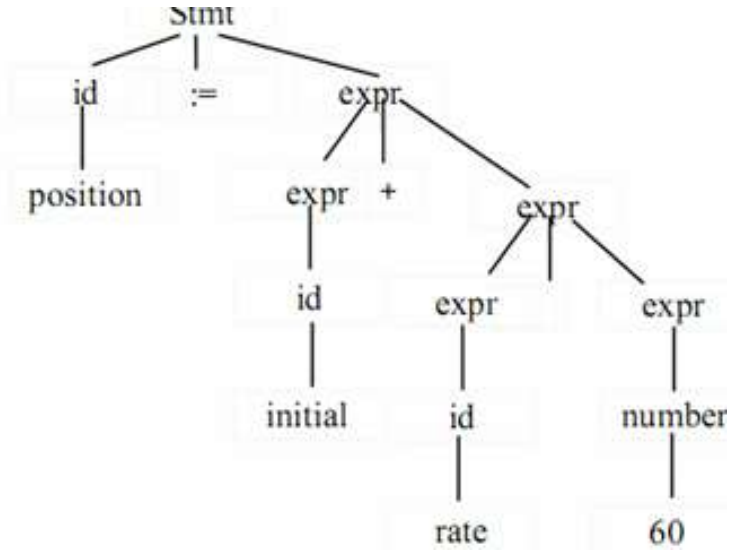
- Phân tích cú pháp dựa vào luật sinh để xây dựng cây phân tích cú pháp.

Ví dụ 1.3: Giả sử ngôn ngữ đặc tả bởi các luật sinh sau :

Stmt → id := expr

expr → expr + expr | expr * expr | id | number

Với câu nhập: position := initial + rate * 60, cây phân tích cú pháp được xây dựng như sau



Hình Một cây phân tích cú pháp

Cấu trúc phân cấp của một chương trình thường được diễn tả bởi quy luật đệ qui.

Ví dụ 1.4:

1) Danh biểu (identifier) là một biểu thức (expr).

2) Số (number) là một biểu thức.

3) Nếu expr1 và expr2 là các biểu thức thì:

expr1 + expr2

expr1 * expr2

(expr)

4)cũng là những biểu thức. Câu lệnh (statement) cũng có thể định nghĩa đệ qui :

1) Nếu id1 là một danh biểu và expr2 là một biểu thức thì id1 := expr2 là một

lệnh (stmt).

2) Nếu expr1 là một biểu thức và stmt2 là một lệnh thì

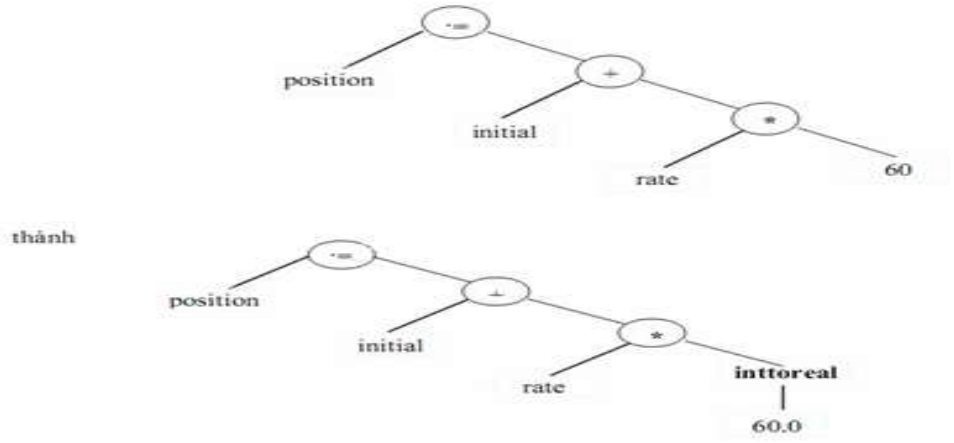
while (expr1) do stmt2 và if (expr1) then stmt2 :đều là các lệnh. Người ta dùng các qui tắc đệ qui như trên để đặc tả luật sinh (production) cho ngôn ngữ. Sự phân chia giữa quá trình phân tích từ vựng và phân tích cú pháp cũng tuỳ theo công việc thực hiện.

1.3.3. Phân tích ngữ nghĩa (Semantic Analysis)

Giai đoạn phân tích ngữ nghĩa sẽ thực hiện việc kiểm tra xem chương trình nguồn có chứa lỗi về ngữ nghĩa hay không và tập hợp thông tin về kiểu cho giai đoạn sinh mã về sau. Một phần quan trọng trong giai đoạn phân tích ngữ nghĩa là kiểm tra kiểu (type checking) và ép chuyển đổi kiểu.

Ví dụ 1.5: Trong biểu thức position := initial + rate * 60

Các danh biểu (tên biến) được khai báo là real, 60 là số integer vì vậy trình biên dịch đổi số nguyên 60 thành số thực 60.0



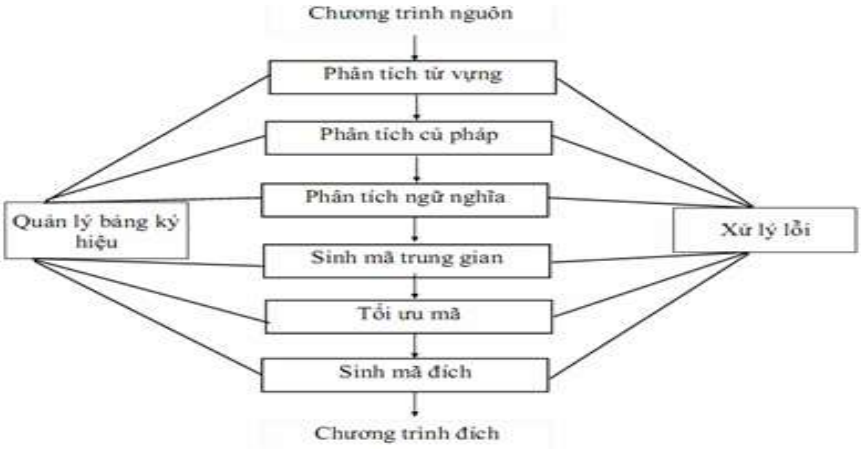
.

Hình Chuyển đổi kiểu trên cây phân tích cú pháp

1.3.4. Các giai đoạn của trình biên dịch

Một trình biên dịch được chia thành các giai đoạn, mỗi giai đoạn chuyển chương trình nguồn từ một dạng biểu diễn này sang một dạng biểu diễn khác.

VÍ DỤ: Một cách phân rã điển hình trình biên dịch được trình bày trong hình



:

Hình Các giai đoạn của một trình biên dịch

Việc quản lý bảng ký hiệu và xử lý lỗi được thực hiện xuyên suốt qua tất cả các giai đoạn. Các giai đoạn mà chúng ta đề cập ở trên là thực hiện theo trình tự logic của một trình biên dịch. Nhưng trong thực tế, cài đặt các hoạt động của nhiều hơn một giai đoạn có thể được nhóm lại với nhau. Thông thường chúng được nhóm thành hai nhóm cơ bản, gọi là: kỳ đầu (Front end) và kỳ sau (Back end).

1. *Kỳ đầu (Front End)*

Kỳ đầu bao gồm các giai đoạn hoặc các phần giai đoạn phụ thuộc nhiều vào ngôn ngữ nguồn và hầu như độc lập với máy đích. Thông thường, nó chứa các giai đoạn sau: Phân tích từ vựng, Phân tích cú pháp, Phân tích ngữ nghĩa và Sinh mã trung gian.Một phần của công việc tối ưu hóa mã cũng được thực hiện ở kỳ đầu. Front end cũng bao gồm cả việc xử lý lỗi xuất hiện trong từng giai đoạn.

2. *Kỳ sau (Back End)*

Kỳ sau bao gồm một số phần nào đó của trình biên dịch phụ thuộc vào máy đích và nói chung các phần này không phụ thuộc vào ngôn ngữ nguồn mà là ngôn ngữ trung gian. Trong kỳ sau, chúng ta gặp một số vấn đề tối ưu hoá mã, phát sinh mã đích cùng với việc xử lý lỗi và các thao tác trên bảng ký hiệu.

1.3.5. Một số phương pháp phân tích cú pháp

1.3.5.1. Topdown

- Phân tích từ trên xuống, từ trái qua phải.
- Khi gặp một từ (terminal) thì phân tích nute tiếp theo.
- Khi không tương ứng với input word thì quay lui.

1.3.5.2. Bottom-up

- Là một dạng của shift-reduce actions.
- Khi gặp vế phải của một luật thì thu gọn thành vế trái.
- Khi không phân tích được tiếp thì quay lui.

1.3.5.3. CYK (Cocke-Younger-Kasami)

- Văn phạm chuẩn Chonsky (Chomsky nomarl Form)

Các luật thuộc một trong hai dạng:

A → B C

A → a

Ví dụ:

S → X Y

X → X A | a | b

Y → A Y | a

A → a

Phân tích: "babaa" →không sinh ra câu

"baaa" →sinh ra câu

S, X			
S, X	S,Y		
S, X	S, X, Y	S, X, Y	
X	X, Y, A	X, Y, A	X, Y, A
b	a	a	a

```
S -> X Y
X -> X A | a | b
Y -> A Y | a
A -> a
"baaa"
```

Xác định các đặc điểm sau đây:

1)Sinh ra giá trị một nút như thế nào?

A[i,j] <- ? + ?

2)Lưu lại đường đi như thế nào để sinh lại cây

Tính nhập nhằng: Một A[,] có thể có nhiều tag, mỗi tag lại được dẫn xuất bằng nhiều cách.

3)Tại sao thuật toán CYK lại cần văn phạm dạng chuẩn Chomsky.

Phân tích câu:

"book that flight"

"book the flight through Houston"

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

Chuyển từ văn phạm CFG sang văn phạm dạng chuẩn Chomsky

1) A -> B C D

A -> X D

X -> B C

2) Bỏ luật dạng A -> B

Với mọi B -> α, sinh luật A -> α

$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Figure 13.8 $\mathscr{L}_1$ Grammar and its conversion to CNF. Note that although they aren't shown here all the original lexical entries from $\mathscr{L}_1$ carry over unchanged as well.

S						
	VP					
S						
	VP			PP		
S		NP			NP	
NP	V, VP	Det	N	P	Det	N
she	eats	a	fish	with	a	fork

Hình. Thử sinh ra một văn pham tương ứng

Thuật toán parsing CYK

function CKY-PARSE(*words*, *grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k],$
          $C \in table[k,j]\}$

Figure 13.10 The CKY algorithm

Đặc điểm

Có thể chuyển mọi văn phạm dạng CFG về dạng chuẩn Chomsky

Searching theo kiểu Bottom-up

Độ phức tạp phân tích là $O(n^3)$

Thuật toán là một dạng của *dynamic programming*

Có thể mở rộng thuật toán CYK để phân tích văn phạm xác suất

1.4. Các ứng dụng của ngôn ngữ tự nhiên

1. Nhận dạng tiếng nói (speech recognition):từ sóng tiếng nói, nhận biết và chuyển chúng thành dữ liệu văn bản tương ứng. Giúp thao tác của con người trên các thiết bị nhanh hơn và đơn giản hơn, chẳng hạn thay vì gõ một tài liệu nào đó bạn đọc nó lên và trình soạn thảo sẽ tự ghi nó ra.Đây cũng là bước đầu tiên cần phải thực hiện trong ước mơ thực hiện giao tiếp giữa con người với robot.Nhận dạng tiếng nói có khả năng trợ giúp người khiếm thị rất nhiều.

2. Tổng hợp tiếng nói (speech synthesis): từ dữ liệu văn bản, phân tích và chuyển thành tiếng người nói .Thay vì phải tự đọc một cuốn sách hay nội dung một trang web, nó tự động đọc cho chúng ta. Giống như nhận dạng tiếng nói, Tổng hợp tiếng nói là sự trợ giúp tốt cho người khiếm thị, nhưng ngược lại nó là bước cuối cùng trong giao tiếp giữa người với robot.

3. Nhận dạng chữ viết (optical character recognition, OCR): từ một văn bản in trên giấy, nhận biết từng chữ cái và chuyển chúng thành một tệp văn bản trên máy tính. có hai kiểu nhận dạng : Thứ nhất là nhận dạng chữ in như nhận dạng chữ trên sách giáo khoa rồi chuyển nó thành dạng văn bản điện tử như dưới định dạng doc của Microsoft Word chẳng hạn.Phức tạp hơn là nhận dạng chữ viết tay, có khó khăn bởi vì chữ viết tay không có khuôn dạng rõ ràng thay đổi từ người này sang người khác.Với chương trình nhận dạng chữ viết in có thể chuyển hàng ngàn đầu sách trong thư viện thành văn bản điện tử trong thời gian ngắn.Nhận dạng chữ viết của con người có ứng dụng trong khoa học hình sự và bảo mật thông tin (nhận dạng chữ ký điện tử).

4. Dịch tự động (machine translation): từ một tệp dữ liệu văn bản trong một ngôn ngữ (tiếng Anh chẳng hạn), máy tính dịch và chuyển thành một tệp văn bản trong một ngôn ngữ khác.Một phần mềm điển hình về tiếng Việt của chương trình này là evtrans của Softex, dịch tự động từ tiếng Anh sang tiếng Việt và ngược lại, phần mềm từng được trang web vdict.com mua bản quyền, đây cũng là trang đầu tiên đưa ứng dụng này lên mạng.Có hai công ty tham gia vào lĩnh vực này cho ngôn ngữ tiếng Việt là công ty Lạc Việt (công ty phát hành từ điển Lạc Việt) và Google

5. Tóm tắt văn bản (text summarization): từ một văn bản dài (mười trang chẳng hạn) máy tóm tắt thành một văn bản ngắn hơn (một trang) với những nội dung cơ bản

6. Tìm kiếm thông tin (information retrieval):

Từ một nguồn rất nhiều tệp văn bản hay tiếng nói, tìm ra những tệp có nội dung liên quan đến một vấn đề (câu hỏi) ta cần biết (hay trả lời].. Điển hình của công nghệ này là Google, một hệ tìm kiếm thông tin trên Web, mà hầu như chúng ta đều dùng thường xuyên. Cần nói thêm rằng mặc dù hữu hiệu hàng đầu như vậy, Google mới có khả năng cho chúng ta tìm kiếm câu hỏi dưới dạng các từ khóa (keywords) và luôn "tìm" cho chúng ta rất nhiều tài liệu không liên quan, cũng như rất nhiều tài liệu liên quan đã tồn tại thì Google lại tìm không ra.

7. Trích chọn thông tin (information extraction):

Từ một nguồn rất nhiều tệp văn bản hay tiếng nói,tìm ra những đoạn bên trong một số tệp liên quan đến một vấn đề (câu hỏi) ta cần biết hay trả lời. Một hệ trích chọn thông tin có thể "lần" vào từng trang Web liên quan, phân tích bên trong và trích ra các thông tin cần thiết, nói gọn trong tiếng Anh để phân biệt với tìm kiếm thông tin là "find things but not pages"

8. Phát hiện tri thức và khai phá dữ liệu văn bản (knowledge discovery and text data mining): Từ những nguồn rất nhiều văn bản thậm chí hầu như không có quan hệ với nhau, tìm ra được những tri thức trước đấy chưa ai biết. Đây là một vấn đề rất phức tạp và đang ở giai đoạn đầu của các nghiên cứu trên thế giới Có thể phân loại các bài toán

- 1-3 thuộc lĩnh vực xử lý tiếng nói và xử lý ảnh (speech and image processing),
- 4-5 thuộc lĩnh vực xử lý văn bản (text processing),
- 6-8 thuộc lĩnh vực khai phá văn bản và Web (text and Web mining).

Chương 2: NGỮ PHÁP TIẾNG ANH

2.1. Các thì trong tiếng anh

- Trong tiếng anh có 12 thì chính, được chia theo điều kiện thời gian như sau:

+ Hiện tại(Present):

- Đơn giản(Simple)
- Tiếp diễn(continuous)
- Hoàn thành(perfect)
- Hoàn thành tiếp diễn(perfect continuous)

+ Quá khứ(Past):

- Đơn giản(Simple)
- Tiếp diễn(continuous)
- Hoàn thành(perfect)
- Hoàn thành tiếp diễn(perfect continuous)

+ Tương lai(Future):

- Đơn giản(Simple)
- Tiếp diễn(continuous)
- Hoàn thành(perfect)
- Hoàn thành tiếp diễn(perfect continuous)

2.2. Cách sử dụng một số thì

2.2.1. Thì hiện tại đơn(The Simple Present Tense):

2.2.1.1 Hịnh thức(Formation)

Thể khẳng định(Affirmative form)

S+ V…(Trong đó S là chủ ngữ, V là động từ thường)

* Nếu chủ ngữ là ngôi thứ 3 số ít(He,She, It, hoặc là một danh từ) thì động từ phải thêm "S" hoặc "ES"

Thể phủ định(Negative form)

S + do not / does not + V…

* "Does not" được sử dụng khi chủ ngữ là ngôi thứ 3 số ít, khi đó động từ ở dạng nguyên thể(không thêm "S" hoặc "ES").

thể nghi vấn(Interrogative form)

Do/Does + s + v…?

**Câu trả lời ngắn:* + Khẳng định: Yes, S + do/does

+Phủ định: No, S + don't/doesn't

2.2.1.2 Cách sử dụng (The uasges)

Diễn tả một sự thật hiển nhiên

Một hành động xảy ra hàng ngày, có tính lặp đi lặp lại

Diễn tả một hành động ở tương lai(thường dùng với các động từ chỉ sự chuyển động như: arrive, leave, return…)

Ex: She leaves tomorrow.

2.2.2. Thì hiện tại tiếp diễn(The present continuous/progressive tense)

2.2.2.1 Hình thức(formation)

Thể khẳng định(Affifmative form)

S + am/is/are + V_ing…

Thể phủ định(Negative form)

S + am not/ is not/ are not + V_ing…

Am not = *'m not, is not = isn't, are not = aren't.

Thể nghi vấn(Interrogative form)

Am/Is/Are + S + V_ing…?

**Câu trả lời ngắn:* +Phẳng định: Yes, S + am/is/are

+Phủ định: No, S + 'm not/isn't/aren't

2.2.2.2 Cách sử dụng(The usages)

Diễn tả một hành động đang sảy ra tại thời điểm nói.

Ex: We are learning English now.

Một hành động xảy ra ở tương lai gần.

Ex: He is watching television tonight.

Một hành động được lặp đi lặp lại nhiều lần, gây bực mình(Thường có trạng từ "always")

Ex: That student is always making noise.

2.2.3. Thì hiện tại hoàn thành(The Present Prefect Tense)

2.2.3.1 Hình thức(Formation)

Thể khẳng định(Affirmative form)

S + have/has + PP… (PP : Quá khứ phân từ)

Have = 've, has = 's

* Nếu chủ ngữ lạ ngôi thứ 3 số ít thì chúng ta dùng "has".

Thể phủ định(Negative form)

S + haven't/ hasn't + PP…

Thể nghi vấn(Interrogative form)

Have/has + S + PP…?

**Câu trả lời ngắn:* +Khẳng định: Yes, S + have/has

+Phủ định: No, S + haven't/hasn't

2.2.3.2 Cách sử dụng(The usages)

Diễn tả một hành động vừa mới xảy ra. Thường có trạng từ "just"

Ex: I have just bought this car.

Diễn tả một hành động xảy ra trong quá khứ không xác định thời gian. Thường có trạng từ "Already"

Ex: He has already read that book.

Diễn tả một hành động bắt đầu ở quá khứ và vẫn còn tiếp tục ở hiện tại.

Các trạng từ chit thời gian thường được dùng: ever, never, so far, since(điểm thời gian), for(khoảng thời gian)…

Ex: I have never driven a car. They have lived here since 1998.

2.2.4. Thì hiện tại hoàn thành tiếp diễn (The Present Prefect continuousTense)

2.2.4.1 Hình thức(Formation)

Thể khẳng định(Affirmative form)

S + have/has + been + V_ing…

Thể phủ định(Negative form)

S + haven't/ hasn't + Been + V_ing…

Thể nghi vấn(Interrogative form)

Have/has + S + Been + V_ing?

**Câu trả lời ngắn:* +Khẳng định: Yes, S + have/has

+Phủ định: No, S + haven't/hasn't

2.2.4.2 Cách sử dụng(The usages)

Diễn tả một hành động bắt đầu còn liên tục đến hiện tại, chấm dứt ở hiện tại hoặc có thể kéo dài đến tương lai.

Ex: I have been waiting for you for a long time.

Lý do xảy ra ngay khi nói.

Ex: Your eyes are very red. Have you been crying?

2.2.5. Thì quá khứ đơn(The Simple Past Tense)

2.2.5.1 Hình thức(Formation)

Thể khẳng định(Affirmative form)

S + V_ed/V2…

* Nếu là động từ có quy tắc thì chúng ta thêm "ED" vào sau động từ thường, nếu là động từ bất quy tắc thì chúng ta sử dụng động từ ở cột 2 trong bảng động từ bất quy tắc.

Thể phủ định(Negative form)

S + did not + V…

did not = didn't

* Khi có trợ động từ "didn't" thì động từ theo sau trở về nguyên thể

Thể nghi vấn(Interrogative form)

Did + S + V…?

* Khi có trợ động từ "Did" thi động từ ở dạng nguyên thể

**Câu trả lời ngắn:* +Khẳng định: Yes, S + did

+Phủ định: No, S + didn't

2.2.5.2 cách sử dụng(The usages)

Diễn tả một hành động xảy ra tại một thời điểm xác định trong quá khứ và đã chấm dứt.

Diễn tả thói quen trong quá khứ.

Ex: She often played badminton when she was young.

Diễn tả các hành động xảy ra kế tiếp nhau trong quá khứ.

Ex: She came in, sat down and said nothing.

2.2.6. Thì quá khứ tiếp diễn (The Past continuous Tense)

2.2.6.1 Hình thức(Formation)

Thể khẳng định(Affirmative form)

S + was/were + V_ing…

Was: dùng cho ngôi I và ngôi thứ 3 số ít.

Thể phủ định(Negative form)

S + was not/ were not + V_ing…

Was not = wasn't, were not = weren't

Thể nghi vấn(Interrogative form)

Was/were + S + V_ing…?

**Câu trả lời ngắn:* +Khẳng định: Yes, S + was/were

+Phủ định: No, S + wasn't/weren't

2.2.6.2 Cách sử dụng (The usages)

Diễn tả một hành động đang diễn ra tại một thời điểm trong quá khứ.

Ex: I was reading book at 8 o'clock last night.

Diễn tả một hành động đang xảy ra ở quá khứ thì bị một hành động khác cắt ngang. Hành động cắt ngang dùng ở thì quá khứ đơn.

Ex: We were watching TV when the light went out.

Một sự việc xảy ra và liên tục trong quá khứ.

Ex: I was sleeping all day yesterdat.

Chỉ 2 hành động xảy ra song song nhau trong quá khứ.

Ex:My father was reading newspaper while my mother was listening to music.

2.2.7. Thì tương lai đơn(The Simple Future Tense)

2.2.7.1 Hình thức(Formation)

Thể khẳng định(Affirmative form)

S + will/shall + V …

* Shall dược dùng cho ngoi I va We. Trong văn nói và trong tiếng anh ngày nay người ta sử dụng "will" cho tất cả các ngôi.

'll: viết tắt của Shall va Will.

Thể phủ định(Negative form)

S + will not/ shall not + V…

will not = won't, shall not = shan't

Thể nghi vấn(Interrogative form)

Will/Shall + S + V…?

**Câu trả lời ngắn:*

+ Khẳng định: Yes, S + will/shall

+ Phủ định: No, S + won't/shan't

2.2.7.2 cách sử dụng (The usages)

Diến tả một hành động sẽ xảy ra tại một thời điểm nào đó trong tương lai.

Ex: She'll be 20 on next Thursaday.

Diễn tả thói quen trong tương lai

Ex: He will go for a walk after dinner.

Diễn tả một việc sẽ quyết định làm ngay lúc nói.

Ex: What would you like to drink? I'll have a mineral water.

Chương 3: CHƯƠNG TRÌNH THỰC NGHIỆM

3.1. Giới thiệu về ngôn ngữ lập trình C#

- C# là một ngôn ngữ lập trình hướng đối tượng được phát triển bởi Microsoft, là phần khởi đầu cho kế hoạch .Nat của họ. Tên của ngôn ngữ bao gồm ký tự thăng (#) theo Microsoft nhưng theo ECMA là C#, chỉ bao gồm dấu số thường. Microsoft phát triển C# dựa trên C++ và Java. C# được miêu tả là ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java. C# được thiết kế chủ yếu bởi Anders Hejlsberg.

- C# là ngôn ngữ rất đơn giản, với khoảng 80 từ khoá và hơn mười kiểu dữ liệu dựng sẵn, nhưng C# có tính diễn đạt cao. C# hỗ trợ lập trình có cấu trúc, hướng đối tượng, hướng thành phần (component oriented).

Trọng tâm của ngôn ngữ hướng đối tượng là lớp. Lớp định nghĩa kiểu dữ liệu mới cho phép mở rộng ngôn ngữ theo hướng cần giải quyết. C# có những từ khoá dành cho việc khai bào lớp, phương thức, thuộc tính (property) mới. C# hỗ trợ đầy đủ khái niệm trụ cột trong lập trình hướng đối tượng: đóng gói, kế thừa, đa hình.

Định nghĩa lớp trong C# không đòi hỏi tách rời tập tin tiêu đề với tập tin cài đặt như C++. Hơn thế, C# hỗ trơ kiểu sưu liệu mới, cho phép sưu liệu trực tiếp trong tập tin mà nguồn. Đến khi biên dịch sẽ tạo tập tin sưu liệu theo dạng XML.

C# hỗ trợ khái niệm giao diện, *interfaces* (tương tự Java). Một lớp chỉ có thể kế thừa duy nhất một lớp cha nhưng có thể cài đặt nhiều giao diện.

C# có kiểu cấu trúc, struct (không giống C++). Cấu trúc là kiểu hạng nhẹ và bị giới hạn. Cấu trúc không thể thừa kế lớp hay được kế thừa nhưng có thể cài đặt giao diện

C# cung cấp những đặc trưng lập trình hướng thành phần như property, sự kiện và dẫn hướng khai báo (được gọi là attribute). Lập trình hướng component được hỗ trợ bởi CLR thông qua siêu dữ liệu (metadata). Siêu dữ liệu mô tả các lớp bao gồm các phương thức và thuộc tính, các thông tin bảo mật…

AssemblyI là một tập hợp các tập tin mà theo cách nhìn của lập trình viên là như các thư viện liên kết động (DLL) hay tập tin thực thi (EXE). Trong .NET một assembly là một đơn vị của việc tái sử dụng, xác định phiên bản, bảo mật và phân phối. CLR cung cấp một số các lớp để thao tác với assembly.

C# cũng cho truy cập trực tiếp bộ nhớ dùng con trỏ kiểu C++, nhưng vùng mã đó được xem là không an toàn. CLR sẽ không thực thi việc thu dọn rác tự động các đối tượng được tham chiếu bởi con trỏ cho đến khi lập trình viên tự giải phóng.

3.1.1. Biến và hằng

3.1.1.1 Biến

- Biến (Variable) là thành phần của một ngôn ngữ lập trình, giúp xử lý dữ liệu một cách linh hoạt và mềm dẻo. Một biến dùng để lưu trữ giá trị mang một kiểu dữ liệu nào đó.

Cú pháp: *[modifier] datatype identifer*;

Với modifier là một trong những từ khoá: public, private, protected,…còn datatype là kiểu dữ liệu (int, long, float,…) và identifer là tên biến.

VD:

public int i;

Ta có thể gán cho biến một giá trị bằng toán tử"=".

i = 10;

Ta cũng có thể khai báo biến và khởi tạo cho biến một giá trị như sau:

int i = 10;

Nếu ta khai báo nhiều biến có cùng kiểu dữ liệu sẽ có dạng:

int x = 10; y = 20;

- Phạm vi hoạt động của biến: là vùng đoạn mã mà từ đấy biến có thể được truy xuất. Trong một phạm vi hoạt động (scope), không thể có hai biến cùng mang một tên trùng nhau

3.1.1.2. Hằng

- Một hằng (constant) là một biến nhưng trị không thể thay đổi được suốt thời gian thi hành chương trình. Đôi lúc ta cũng cần có những giá trị bao giờ cũng bất biến

VD:

Const int i = 10; // giá trị này không thể bị thay đổi

Trong định nghĩa lớp người ta thường định nghĩa những mục tin (field) được gọi là read-only variable, nghĩa là những biến chỉ được đọc mà thôi.

- Hằng có những đặc điểm:

+ Hằng bắt buộc phải được gán giá trị lúc khai báo. Một khi đã được khởi gán thì không thể viết đè chồng lên.

+ Trị của hằng có thể được tính toán vào lúc biên dịch. Do đó không thể gán một hằng từ một giá trị của biến. Nếu muốn làm thế thì phải sử dụng đến một read-only field.

+ Hằng bao giờ cũng static, tuy nhiên ta không thể đưa từ khoá static vào khi khai báo hằng.

- Thuận lợi khi sử dụng hằng:

+ Hằng làm cho chương trình đọc dễ dàng hơn bằng cách thay thế những con số vô cảm bởi những tên mang đầy đủ ý nghĩa hơn.

+ Hằng làm cho dễ sửa chữa chương trình hơn.

+ Hằng làm cho việc tránh lỗi dễ dàng hơn, nếu gán một giá trị khác cho một hằng đâu đó trong chương trình sau khi đã gán giá trị cho hằng, thì trình biên dịch sẽ thông báo lỗi.

3.1.2. Dữ liệu kiểu trị và kiểu qui chiếu

- C# là một ngôn ngữ được kiểm soát chặt chẽ về mặt kiểu dữ liệu, ngoài ra C# còn chia các kiểu dữ liệu thành hai loại khác nhau: *kiểu trị* (value type) và *kiểu qui chiếu* (reference). Nghĩa là trên một chương trình C# dữ liệu được lưu trữ một hoặc hai nơi tuỳ theo đặc thù của kiểu dữ liệu.

+ Chỗ thứ nhất là *stack* một vùng ký ức dành lưu trữ dữ liệu chiều dài cố định, chẳng hạn int chiếm dụng 4 byte. Mỗi chương trình khi đang thi hành đều được cấp phát riêng một stack riêng biệt mà các chương trình khác ko được động tới. Khi một hàm được gọi hàm thi hành thì tất cả các biến cục bộ của hàm được ấn vào stack và khi hàm hoàn thành công tác thì những biến cục bộ của hàm đều bị tống ra. Đây là cách thu hồi khi hàm khi hết hoạt động.

+ Chỗ thứ hai là *heap,* một vùng ký ức dùng lưu trữ dữ liệu có bề dày thay đổi và khá đồ sộ, string chẳng hạn, hoặc dữ liệu có một cuộc sống dài hơn phương thức của một đối tượng chẳng hạn.

VD: Khi phương thức thể hiện (instantiate) một đối tượng, đối tượng được lưu trữ trên heap, và nó không bị đẩy ra khi hàm hoàn thành giống như stack, mà ở nguyên tại chỗ và có thể trao cho các phương thức khác thông qua một quy chiếu. Trên C# heap này được gọi là *managed heap*. C# cũng hỗ trợ kiểu con trỏ (pointer type) giống như C++ nhưng ít khi dùng đến và chỉ dùng khi làm việc với đoạn mã unmanaged.

3.1.2.1. Kiểu giá trị được định nghĩa trước (Predefined Value Types)

-Các kiểu interger:

Name	CTS Type	Description	Range (min:max)
sbyte	System.Sbyte	8-bit signed interger	-128:127 (-2^7:2^7-1)
short	System.Int16	16-bit signed interger	-32,768:32,767 (-2^{15}:2^{15}-1)
int	System.Int32	32-bit signed interger	-2,147,483,648: 2,147,483,647 (-2^{31}:2^{31}-1)
long	System.Int64	64-bit signed interger	-9,223,372,036,854,775,808: 9,223,372,036,854,775,807 (-2^{63}:2^{63}-1)
byte	System.Byte	8-bit signed interger	0:255 (0:2^8-1)
ushort	System.UInt16	16-bit signed interger	0:65,535 (0:2^{16}-1)
uint	System.Uint32	32-bit signed interger	0:4,294,967,295 (0:2^{32}-1)
ulong	System.Uint64	64-bit signed interger	0:18,446,744,073,709,551,615(0:2^{64}-1)

- Kiểu dữ liệu số dấu chấm di động (Floating Point Types)

Name	CTS Type	Description	Significant Figures	Range (approximate)
Float	System.Single	32-bit single-precision floating-point	7	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$
Double	System.Double	64-bit double-precision floating-point	15/16	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$

- Kiểu dữ liệu số thập phân (Decimal Type)

Name	CTS Type	Description	Significant Figures	Range (approximate)
decimal	System.Decimal	128-bit high precision decimal notation	28	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$

- Kiểu Boolean

Name	CTS Type	value
Bool	System.Boolean	True or false

- Kiểu Character Type

Name	CTS Type	Value
char	System.Char	Represents a single 16-bit (Unicode) character

3.1.2.2. Kiểu tham khảo tiền định nghĩa

- Các kiểu chuỗi: Đối tượng kiểu string thường chứa một chuỗi ký tự. Khi khai báo một biến chuỗi sử dụng từ khoá string giống như sau:

```
string    myString;
```

Thường thì khởi gán một biến chuỗi sử dụng đến một kiểu string:

```
string   myString = “Xin chao” ;
string str1 = “Hello”;
string str2 = “World”;
string str3 = str1 + str2;
```

3.1.3.Câu lệnh điều kiện

3.1.3.1. Câu lệnh điều kiện

- Câu lệnh điều kiện if:

Cú pháp:

```
if   (condition)
     statement(s)
[else
     statement(s)]
```

VD:

```
bool isZero;
if (i == 0)
{
  isZero = true;
  Console.WriteLine("i is Zero");
}
else
{
   isZero = false;
   Console.WriteLine("i is Non-zero");
}
```

Đoạn code trên kiểm tra isZero có bằng 0 hay không.

- Câu lệnh switch:

Cú pháp:

```
switch (biểu thức)
{   casce biểu thức ràng buộc:
    câu lệnh
     câu lệnh nhảy
    [default: câu lệnh mặc định]
 }
```

VD:

```
//assume country and language are of type string
 switch(country)
   {
```

```
            case "America":
                CallAmericanOnlyMethod();
                goto case "Britain";
            case "France":
                language = "French";
                break;
            case "Britain":
              language = "English";
              break;
        }
    switch(country)
      {
          case "au":
          case "uk":
          case "us":
              language = "English";
              break;
          case "at":
          case "de":
             language = "German";
             break;
        }

        }
```

3.1.3.2. Vòng lặp (Loops)

- Vòng lặp for:

Cú pháp:

for *(initializer; condition; iterator)*

statement(s)

VD:

```
    for (int i = 0; i < 100; i = i+1)
      {
         Console.WriteLine(i);
```

```
}
```

- Vòng lặp While:

Ý nghĩa: Trong khi điều kiện đúng thì thực hiện các công việc này

Cú pháp:

```
While(condition)
    statement(s);
```

VD:

```
bool condition = false;
while (!condition)
{
  // Vòng lặp thực hiện đến khi điều kiện đúng
    DoSomeWork();
    condition = CheckCondition(); //cho rằng CheckCondition() trả về kiểu  bool
}
```

Biểu thức của vòng lặp while là điều kiện để các lệnh được thực hiện, biểu thức này bắt buộc phải trả về một giá trị kiểu bool là true/false. Nếu có nhiều câu lệnh cần được thực hiện trong vòng lặp while thì phải đặt các lệnh này trong khối lệnh.

- Vòng lặp do……while:

```
bool condition;
do
    {
    // Vòng lặp này sẽ thực hiện ít nhất một lần thậm chí nếu câu điều kiện sai
        MustBeCalledAtLeastOnce();
        condition = CheckCondition();
    } while (condition);
```

- Vòng lặp foreach:

Vòng lặp forech cho phép tạo vòng lặp thông qua một tập hợp hay một mảng. Đây là một câu lệnh lặp mới không có trong ngôn ngữ C/C++.

Cú pháp:

foreach (type identifier in expression) statement

(hay: foreach (<kiểu tập hợp> <tên truy cập thành phần> in <tên tập hợp><các câu lệnh thực hiện>)).

VD:

```
foreach (int temp in arrayOfInts)
  {
     Console.WriteLine(temp);
   }


 foreach (int temp in arrayOfInts)
   {
     temp++;
     Console.WriteLine(temp);
   }
```

Do lặp dựa trên một mảng hay tập hợp nên toàn bộ vòng lặp sẽ duyệt qua tất cả các thành phần cảu tập hợp theo thứ tự được sắp. Khi duyệt đến phần tử cuối cùng trong tập hợp thì chương trình sẽ thoát ra khỏi vòng lặp foreach.

- Câu lệnh goto:

Cú pháp:

```
goto Label1;
Console.WriteLine("This won't be executed");
Label1:
Console.WriteLine("Continuing execution from here");
```

- Câu lệnh break:

Ta dùng câu lệnh break khi ta muốn ngưng ngang việc thi hành và thoát khỏi vòng lặp

- Câu lệnh continue:

Câu lệnh continue được dùng trong vòng lặp khi bạn muốn khởi động lại vòng lặp nhưng lại không muốn thi hành phần lệnh còn lại trong vòng lặp, ở một điểm nào đó trong thân vòng lặp.

- Câu lệnh return:

Câu lệnh return dùng thoát khỏi một hàm hành sự của một lớp, trả quyền điều khiển về phía triệu gọi hàm (caller). Nếu hàm có một kiểu dữ liệu trả về thì return phải trả về một kiểu dữ liệu này; bằng không thì câu lệnh được dùng không có biểu thức.

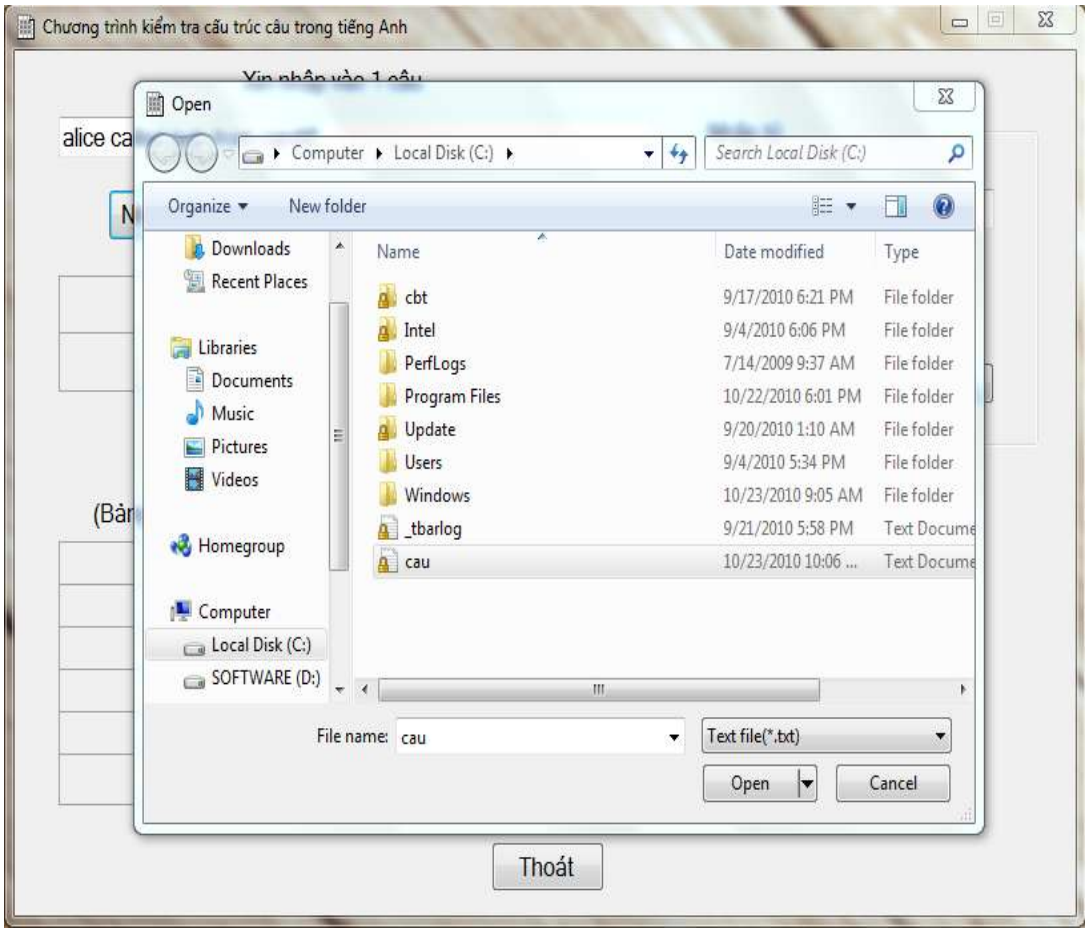
3.2. Chương trình

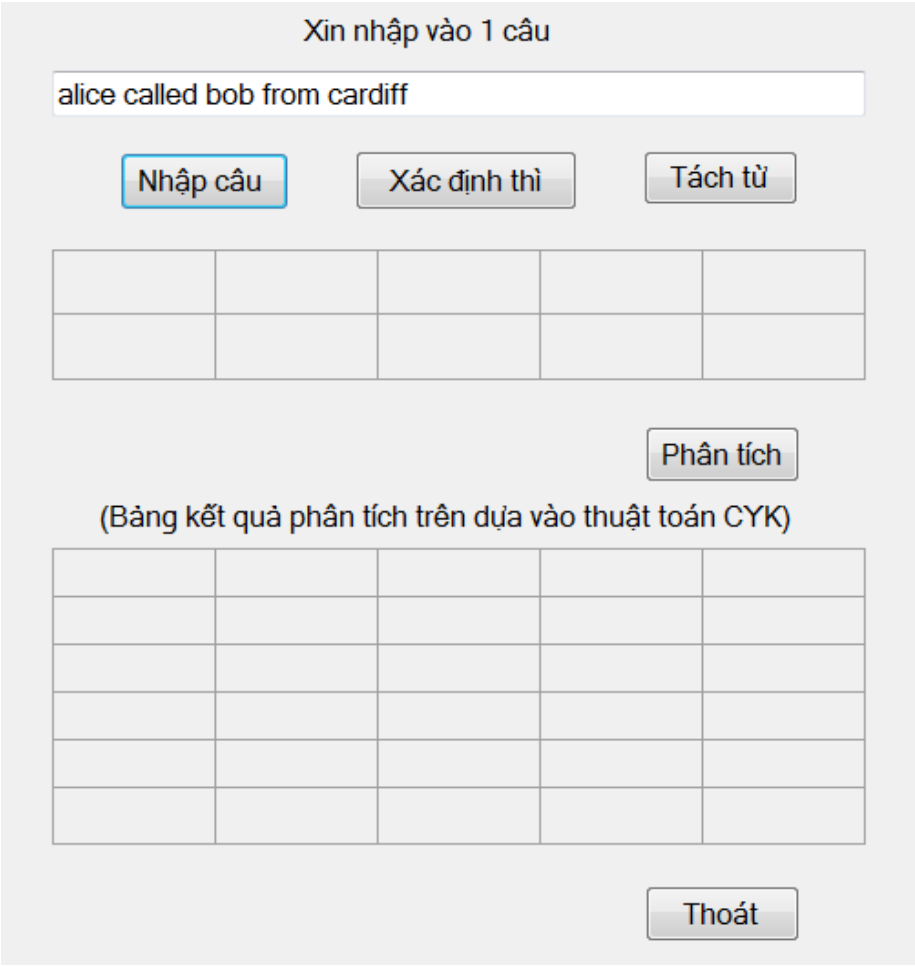
Chương trình sẽ được chạy theo nhiều bước:

- Nhập câu cần kiểm tra.
- Tách câu thành từng từ (chữ).
- Gán nhãn cho từng từ (chữ) vừa tách.
- Sử dụng thuật toán CYK để phân tích cú pháp và đưa ra cây cú pháp của câu.

3.2.1. Nhập câu từ bàn phím

Câu có thể được nhập trực tiếp từ bàn phím, hoặc đọc từ một file text đã lưu sẵn. Khi muốn nhập câu từ file text ta bấm vào nút "Nhập câu", một cửa sổ hiện ra cho phép ta tìm đến file text mà ta đã lưu trữ. Bấm chuột trái vào file cần mở sau đó bấm Open để mở file.





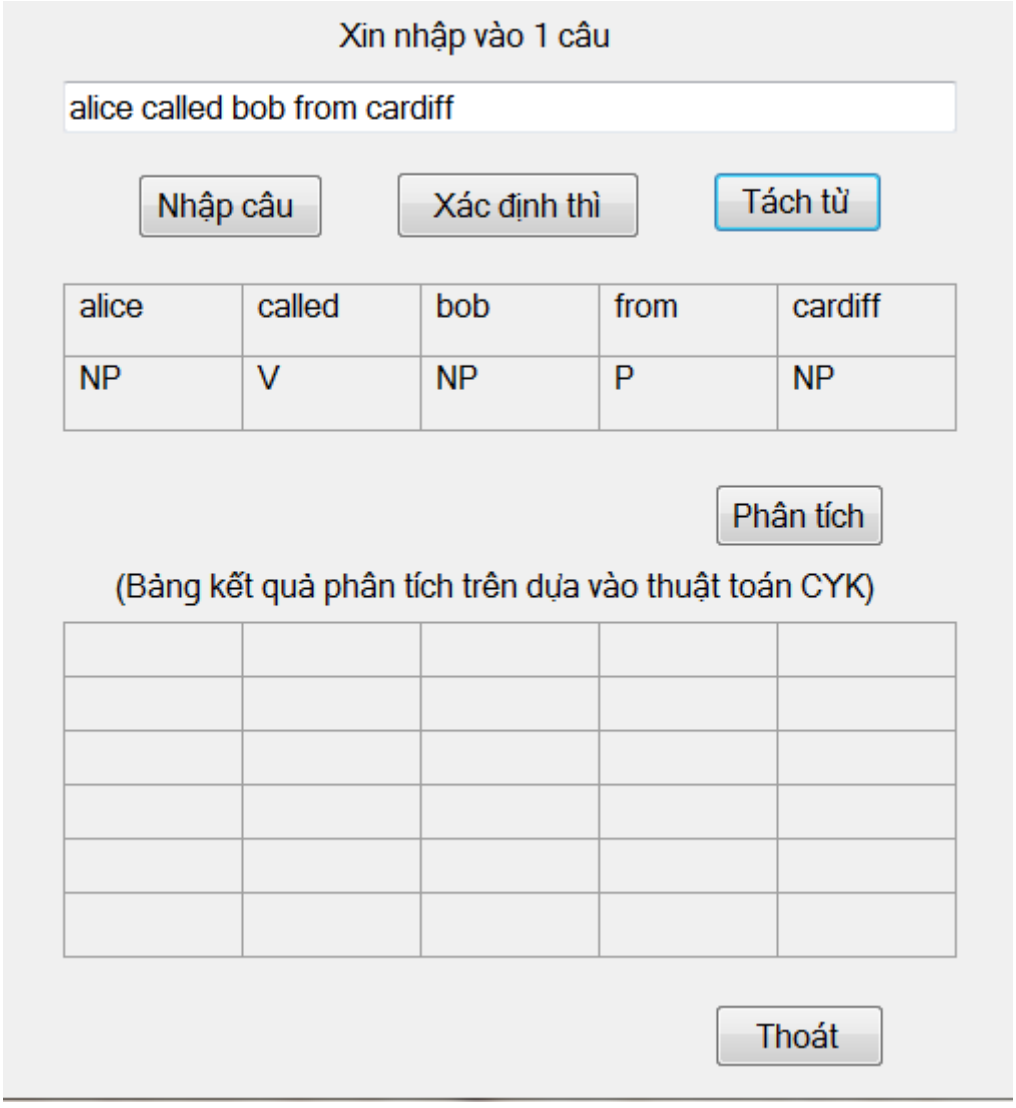
Câu lệnh:

```
        private void btnNhapcau_Click(object sender, EventArgs e)
{
    Stream myStream = null;
    OpenFileDialog openFile = new OpenFileDialog();
    openFile.InitialDirectory = "C:\\";
    openFile.Filter = "Text file(*.txt)|*txt|All files(*.*)|*.*";
    openFile.RestoreDirectory = true;
    if (openFile.ShowDialog() == DialogResult.OK)
    {
        try
        {
            if ((myStream = openFile.OpenFile()) != null)
            {
                using (myStream)
                {
                    using (StreamReader sr = new
                                StreamReader(openFile.FileName))
                    {
```

```
                        txtNhapcau.Text = sr.ReadLine();
                    }
                }
            }
        }
        catch (Exception ea)
        {
            MessageBox.Show("Không mở được file" +
                                        ea.ToString());
        }
    }
}
```

3.2.2.Tách từ và gán nhãn

Bấm chuột nút "Tách từ" chương trình sẽ thực hiện việc tách các từ của câu đồng thời gán nhãn từ loại cho các từ vừa tách.



Câu lệnh:

```
            private void btnTachtu_Click(object sender, EventArgs e)
    {
        if (txtNhapcau.Text == "")
        {
            MessageBox.Show("Không có câu được nhập");
        }
        else
        {
            string[] cauNhap = txtNhapcau.Text.Split();
            int length = cauNhap.Length;

            for (int i = 0; i < length; i++)
            {
                if (cauNhap[i].StartsWith("a"))
                {
                    DuyetTu("a.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("b"))
                {
                    DuyetTu("b.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("c"))
                {
                    DuyetTu("c.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("d"))
                {
                    DuyetTu("d.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("e"))
                {
                    DuyetTu("e.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("f"))
                {
                    DuyetTu("f.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("g"))
                {
                    DuyetTu("g.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("h"))
                {
                    DuyetTu("h.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("k"))
                {
                    DuyetTu("k.txt", cauNhap[i], i);
```

```
            }
            if (cauNhap[i].StartsWith("l"))
            {
                DuyetTu("l.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("m"))
            {
                DuyetTu("m.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("n"))
            {
                DuyetTu("n.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("i"))
            {
                DuyetTu("i.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("j"))
            {
                DuyetTu("j.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("o"))
            {
                DuyetTu("o.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("p"))
            {
                DuyetTu("p.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("q"))
            {
                DuyetTu("q.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("r"))
            {
                DuyetTu("r.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("s"))
            {
                DuyetTu("s.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("t"))
            {
                DuyetTu("t.txt", cauNhap[i], i);
            }
            if (cauNhap[i].StartsWith("v"))
            {
                DuyetTu("v.txt", cauNhap[i], i);
            }
```

```
                if (cauNhap[i].StartsWith("u"))
                {
                    DuyetTu("u.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("y"))
                {
                    DuyetTu("y.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("z"))
                {
                    DuyetTu("z.txt", cauNhap[i], i);
                }
                if (cauNhap[i].StartsWith("w"))
                {
                    DuyetTu("w.txt", cauNhap[i], i);
                }
            }
        }
    }

    public void DuyetTu(string tudien, string cauNhap, int i)
    {
        try
        {
            using (StreamReader sr = new StreamReader(tudien))
            {
                string line;
                while ((line = sr.ReadLine()) != null)
                {
                    string[] dong = line.Split();
                    if (cauNhap == dong[0])
                    {
                        Label label1 = new Label();
                        label1.Text = dong[0];
                        tlpTachtu.Controls.Add(label1, i, 0);

                        Label label2 = new Label();
                        label2.Text = dong[1];
                        tlpTachtu.Controls.Add(label2, i, 1);
                    }
                }
            }
        }
        catch (Exception a)
        {
            MessageBox.Show("Không đọc được file", a.ToString());
        }
    }
```

3.2.3. Bảng phân tích cú pháp

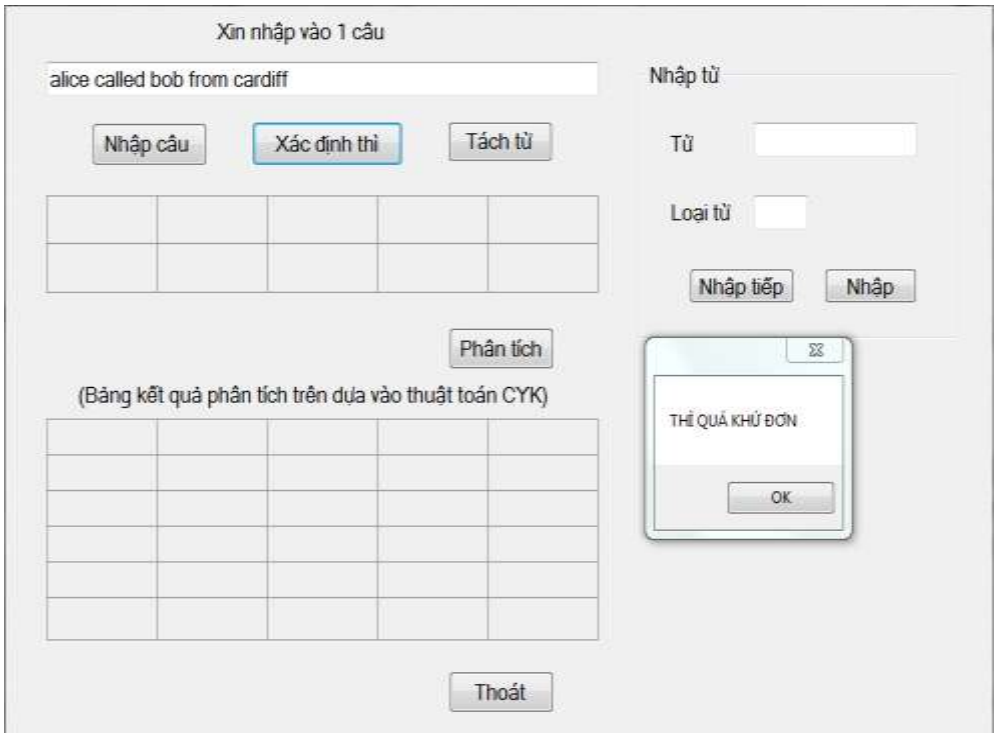
Sử dụng thuật toán CYK để phân tích cú pháp. Bấm chuột vào nút "Phân tích" chương trình sẽ tiến hành phân tích cú pháp và đưa ra bảng kết quả phân tích. Kết quả như hình dưới.

Đoạn chương trình của giải thuật CYK:

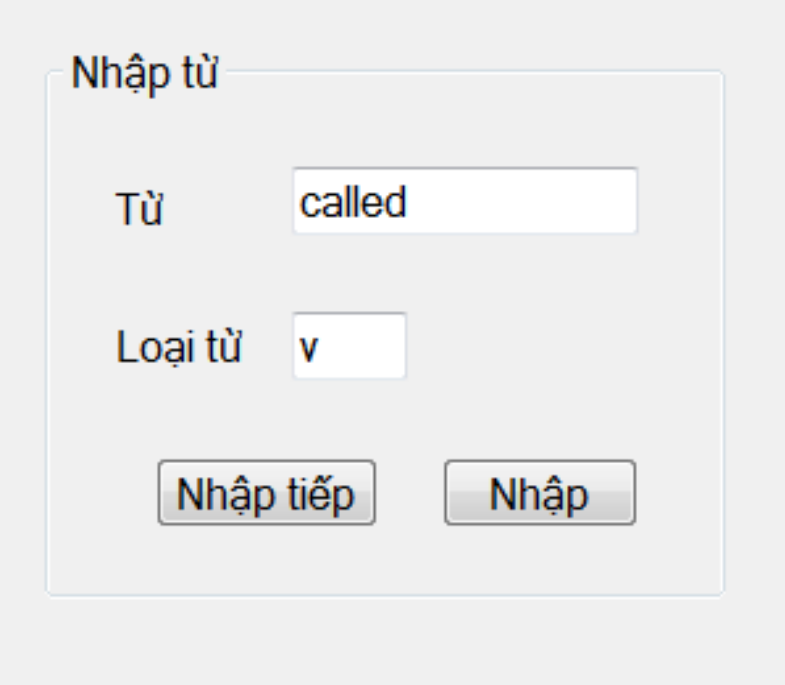
```
for (int j = 3; j < length + 1; j++)
 {
     for (int i = 0; i < length - j + 1; i++)
     {
         for (int k = 1; k < j - 1; k++)
         {
             string t = KiemtraCYK(CYKTable[i, k], CYKTable[i
                                                 + k, j - k]);
             if (t !="")
             {
                 CYKTable[i, j] = t;
             }
         }
     }
 }
 return CYKTable;
```

3.2.4. Kiểm tra thì của câu

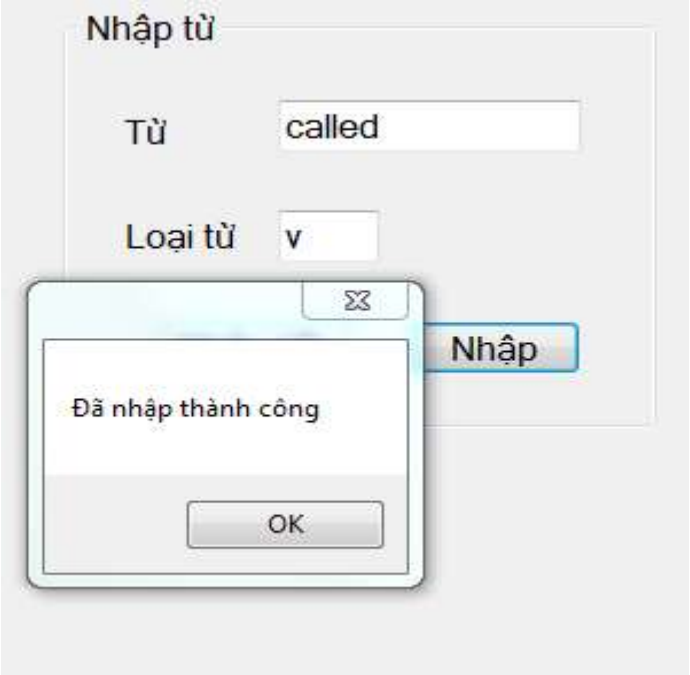
Khi câu được nhập vào, bấm chuột vào nút "Xác định thì" chương trình sẽ cho ta biết câu được dùng ở thì nào.



3.2.5. Nhập thêm từ vào bộ từ điển



Khi kiểm tra một câu mà các từ của câu đó không có trong từ điển ta có thế nhập thêm từ đó vào bộ từ điển. (Hình trên) Một ô nhập vào từ cần nhập, ô còn lại là loại từ của từ đó trong câu. Bấm chuột vào nút "Nhập" để tiến hành nhập.



Sau khi nhập thành công sẽ có thông báo: "Đã nhập thành công" (như hình trên)

Câu lệnh:

```
public void NhapTu(string filename, string text, string type)
{
    try
    {
        using(StreamWriter sw = new StreamWriter(filename, true))
        {
            sw.WriteLine("\n" + text + " " + type);
            MessageBox.Show("Đã nhập thành công");
            txtTu.Text = "";
            txtLoai.Text = "";
        }
    }
    catch(Exception loi)
    {
        MessageBox.Show("Không mở được file" + loi.Message);
    }
}
```

```
private void btnNhap_Click(object sender, EventArgs e)
{
    string word;
    string loai;
    word = txtTu.Text;
    loai = txtLoai.Text.ToUpper();
    if (word.StartsWith("a"))
    {
        NhapTu("a.txt", word, loai);
    }
    if (word.StartsWith("b"))
    {
        NhapTu("b.txt", word, loai);
    }
    if (word.StartsWith("c"))
    {
        NhapTu("c.txt", word, loai);
    }
    if (word.StartsWith("d"))
    {
        NhapTu("d.txt", word, loai);
    }
    if (word.StartsWith("e"))
    {
        NhapTu("e.txt", word, loai);
    }
    if (word.StartsWith("f"))
    {
        NhapTu("f.txt", word, loai);
    }
    if (word.StartsWith("g"))
    {
        NhapTu("g.txt", word, loai);
    }
    if (word.StartsWith("h"))
    {
        NhapTu("h.txt", word, loai);
    }
    if (word.StartsWith("i"))
    {
        NhapTu("i.txt", word, loai);
    }
```

```
            if (word.StartsWith("j"))
            {
                NhapTu("j.txt", word, loai);
            }
            if (word.StartsWith("k"))
            {
                NhapTu("k.txt", word, loai);
            }
            if (word.StartsWith("l"))
            {
                NhapTu("l.txt", word, loai);
            }
            if (word.StartsWith("m"))
            {
                NhapTu("m.txt", word, loai);
            }
            if (word.StartsWith("n"))
            {
                NhapTu("n.txt", word, loai);
            }
            if (word.StartsWith("o"))
            {
                NhapTu("o.txt", word, loai);
            }
            if (word.StartsWith("p"))
            {
                NhapTu("p.txt", word, loai);
            }
            if (word.StartsWith("q"))
            {
                NhapTu("q.txt", word, loai);
            }
            if (word.StartsWith("r"))
            {
                NhapTu("r.txt", word, loai);
            }
            if (word.StartsWith("s"))
            {
                NhapTu("s.txt", word, loai);
            }
            if (word.StartsWith("t"))
            {
                NhapTu("t.txt", word, loai);
```

```
            }
            if (word.StartsWith("u"))
            {
                NhapTu("u.txt", word, loai);
            }
            if (word.StartsWith("y"))
            {
                NhapTu("y.txt", word, loai);
            }
            if (word.StartsWith("v"))
            {
                NhapTu("v.txt", word, loai);
            }
            if (word.StartsWith("w"))
            {
                NhapTu("w.txt", word, loai);
            }
            if (word.StartsWith("x"))
            {
                NhapTu("x.txt", word, loai);
            }
            if (word.StartsWith("z"))
            {
                NhapTu("z.txt", word, loai);
            }
        }
```

KẾT LUẬN

1. Về lý thuyết

Tìm hiểu cơ sở khoa học của ngôn ngữ tự nhiên, lý thuyết về thông tin và một số ứng dụng của nó.

- Trình bày tổng quan về xử lý ngôn ngữ tự nhiên, quy trình sử lý ngôn ngữ tự nhiên và các ứng dụng của nó.
- Một số phương pháp phân tích cú pháp
- Ngữ pháp tiếng Anh cơ bản.

2. Thực nghiệm (Chương trình mô phỏng)

Chương trình dừng lại ở mô phỏng khái quát việc kiểm tra cấu trúc câu trong tiếng Anh. Chỉ kiểm tra được những câu ngắn (tối đa 5 chữ cái) và có cấu trúc đơn giản.

Hướng phát triển: Kiểm tra được một câu bất kỳ, xây dựng chương trình kiểm tra cấu trúc câu trong tiếng Việt và một số chương trình dịch tự động,…

TÀI LIỆU THAM KHẢO

1. TS. Lê Anh Cường, Bài giảng Xử lý ngôn ngữ tự nhiên, Khoa CNTT, Đại học Công nghệ - Đại học Quốc gia Hà Nội, 2007.

2. Christopher D. Manning, Hinrich Schütze, Foundations of Statistical Natural Language Processing, The MIT Press Cambridge Massachusetts London England, (p32-55) 1999.

3. Steve Renals, Probabilistic context-free grammars, Lecture,11- 2005.

4. http://en.wikipedia.org/wiki/CYK_algorithm

5. Internet