TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

ĐỀ TÀI: **PHÂN CỤM VĂN BẢN**

Nhóm 7

Sinh viên thực hiện: Đoàn Quang Trung : 0076767
Phan Văn Hiếu : 0178267
Mai Quang Nguyên : 0030667
Nguyễn Bá Minh : 0038767

Giảng viên bộ môn: Thầy Nguyễn Đình Quý

Hà Nội, 05-2025

MỤC LỤC

LỜI CẢM ƠN

Trong quá trình thực hiện báo cáo của học phần Xử lý ngôn ngữ tự nhiên với đề tài Phân cụm văn bản, chúng em xin chân thành gửi lời cảm ơn tới các thầy cô là giảng viên các bộ môn trong khoa Công nghệ Thông tin – Trường Đại học Xây Dựng Hà Nội đã truyền đạt cho chúng em những kiến thức nền tảng liên quan đến học sâu và trí tuệ nhân tạo. Đồng thời, chúng em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Đình Quý, người đã trực tiếp giảng dạy và hỗ trợ chúng em trong suốt quá trình lên ý tưởng, nghiên cứu và hoàn thiện báo cáo này. Sự chỉ dẫn, hướng dẫn và những kiến thức quý báu mà thầy đã truyền đạt cho sinh viên trong suốt quá trình thực hiện báo cáo cũng như trong quá trình học. Sự kiên nhẫn và tận tâm của thầy đã giúp sinh viên như chúng em vượt qua những khó khăn và hoàn thiện báo cáo một cách tốt nhất.

Chúng em cũng muốn bày tỏ lòng biết ơn đến khoa Công nghệ thông tin trường Đại học Xây Dựng Hà Nội đã tạo điều kiện thuận lợi và cung cấp những kiến thức chuyên môn quan trọng cho chúng em trong suốt quá trình học tập và nghiên cứu tại trường.

Trong quá trình thực hiện nghiên cứu đề tài, do năng lực, kiến thức cũng như kỹ năng của chúng em còn chưa được hoàn thiện, thiếu chuyên sâu nên không thể tránh khỏi những sai sót về kỹ thuật. Vì vậy, chúng em rất muốn được lắng nghe những góp ý từ các thầy cô để từ đó rút kinh nghiệm và hoàn thành bài báo cáo này một cách tốt nhất. Chúng em xin chân thành cảm ơn.

Chúng em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Sự phát triển không ngừng của Công nghệ thông tin đã mở ra một cánh cửa mới cho việc hiểu và khai thác thông tin, đặc biệt là trong lĩnh vực xử lý văn bản. Việc tận dụng công nghệ để phân cụm văn bản không chỉ là một tiến bộ vượt bậc trong lĩnh vực này mà còn mở ra những cơ hội đáng kể trong việc tổ chức, tìm kiếm và trích xuất thông tin quan trọng từ nguồn dữ liệu vô cùng phong phú.

Trí tuệ nhân tạo (AI) đã đóng vai trò quan trọng trong việc tiếp cận và hiểu biết sâu rộng hơn về văn bản. Việc áp dụng kỹ thuật phân cụm vào văn bản không chỉ giúp chúng ta hiểu rõ hơn về cấu trúc, ngữ cảnh và nội dung của từng đoạn văn mà còn giúp tạo ra những phân khúc thông tin rõ ràng, giúp định hình và tối ưu hóa quá trình tìm kiếm, xử lý và trình bày dữ liệu.

Đề tài này hướng đến việc sử dụng các phương pháp phân cụm văn bản để hiểu sâu hơn về cấu trúc và mối liên hệ giữa các đoạn văn, tạo ra các nhóm văn bản có tính tương đồng về nội dung hoặc ngữ cảnh. Việc này không chỉ giúp tối ưu hóa quá trình xử lý thông tin mà còn mở ra những cơ hội mới trong việc khám phá và ứng dụng thông tin từ các nguồn văn bản phong phú.

Trong báo cáo này, chúng ta sẽ cùng nhau khám phá và đánh giá những kỹ thuật, phương pháp phân cụm văn bản và nhìn nhận về tiềm năng của chúng trong việc hiểu và tận dụng thông tin từ các nguồn văn bản khác nhau. Chúng ta sẽ xem xét từ việc nắm bắt cơ bản về bài toán đến việc áp dụng thực tiễn thông qua các kỹ thuật phân cụm, và kết quả thu được thông qua các thử nghiệm và đánh giá.

Mục tiêu không chỉ đơn thuần là tạo ra một hệ thống phân cụm văn bản hiệu quả mà còn là tạo ra một cơ sở kiến thức vững chắc về phương pháp này, đồng thời khai thác tiềm năng của nó trong việc tổ chức và trích xuất thông tin từ văn bản.

Bố cục của đề tài bao gồm các chương chính như sau:

CHƯƠNG I. GIỚI THIỆU VỀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

1. Tổng quan

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực nghiên cứu và ứng dụng của trí tuệ nhân tạo liên quan đến tương tác giữa con người và máy tính thông qua ngôn ngữ tự nhiên. Mục tiêu chính của NLP là giúp máy tính hiểu, diễn giải và tạo ra ngôn ngữ tự nhiên một cách tự động.

NLP bao gồm nhiều phần tử và công nghệ khác nhau, bao gồm:

- **Xử lý ngôn ngữ tự nhiên**: Đây là quá trình xử lý và phân tích ngôn ngữ tự nhiên. Điều này bao gồm việc tách từ, phân tích cú pháp, phân loại ngữ nghĩa, trích xuất thông tin và truy vấn ngôn ngữ tự nhiên.

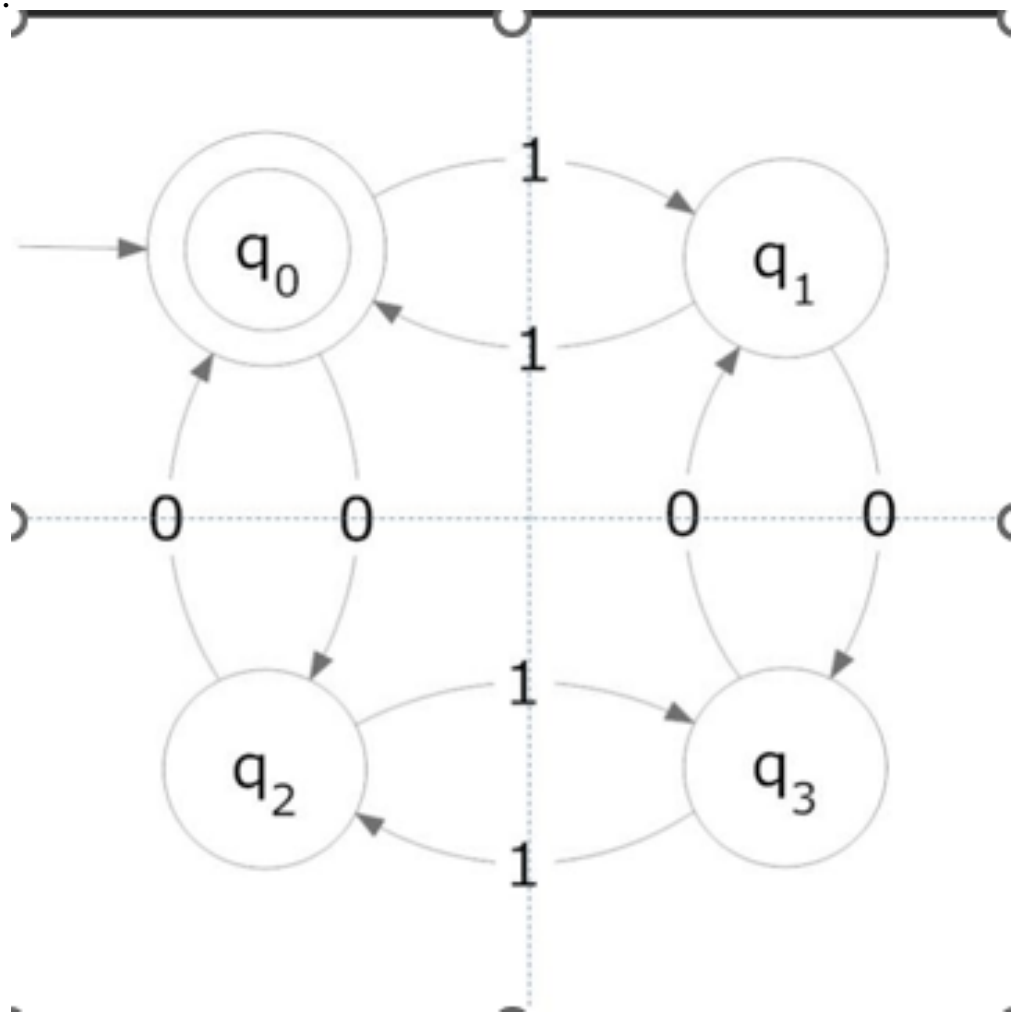
- **Hiểu ngôn ngữ tự nhiên**: Mục tiêu là giúp máy tính hiểu ý nghĩa của ngôn ngữ tự nhiên. Điều này bao gồm hiểu ý nghĩa của câu, xác định ngữ cảnh, nhận dạng tác giả hoặc người nói, và hiểu ý đồ của người dùng.

- **Tạo ngôn ngữ tự nhiên**: Đây là quá trình tạo ra ngôn ngữ tự nhiên từ dữ liệu không phải ngôn ngữ tự nhiên. Ví dụ, tạo ra câu mô tả từ dữ liệu số hoặc tạo ra bài viết tự động từ dữ liệu cấu trúc.

- **Dịch máy**: Dịch máy là quá trình chuyển đổi văn bản từ một ngôn ngữ sang ngôn ngữ khác một cách tự động. Dịch máy đã có sự phát triển đáng kể với sự xuất hiện của các mô hình học sâu như Transformers.

- **Học máy trong NLP**: Học máy đóng vai trò quan trọng trong NLP. Các phương pháp học máy, bao gồm học có giám sát và học không giám sát, được sử dụng để xây dựng các mô hình NLP có khả năng học và hiểu ngôn ngữ tự nhiên.

Ứng dụng của NLP rộng rãi và đa dạng. Chúng có thể được sử dụng trong hệ thống tìm kiếm thông tin, chatbot, giao diện người-máy, phân tích ý kiến, tổ chức và tóm tắt văn bản, dò tìm tri thức và nhiều lĩnh vực khác.



Hình 1: Tiền đề trong việc xây dựng lý thuyết Automata là ngôn ngữ hình thức

NLP đã có những tiến bộ đáng kể trong thập kỷ gần đây, đặc biệt là nhờ vào sự phát triển của các mô hình học sâu và tập dữ liệu lớn. Tuy nhiên, vẫn còn nhiều thách thức trong NLP, bao gồm khả năng hiểu ngữ cảnh, xử lý ngôn ngữ không chuẩn và đa nghĩa, và đảm bảo tính công bằng và an toàn trong việc sử dụng công nghệ NLP.

2. Cơ sở khoa học

Cơ sở khoa học của Xử lý Ngôn ngữ Tự nhiên (Natural Language Processing - NLP) dựa trên nhiều lĩnh vực và nguyên tắc trong khoa học máy tính và ngôn ngữ học. Dưới đây là một số cơ sở khoa học quan trọng trong NLP:

1. **Ngôn ngữ học**: NLP dựa trên các nguyên tắc và kiến thức về ngôn ngữ học. Ngôn ngữ học nghiên cứu về cấu trúc và chức năng của ngôn ngữ, bao gồm cú pháp, ngữ nghĩa, và ngữ âm. Ngôn ngữ học cung cấp các khái niệm và phương pháp để phân tích và hiểu ngôn ngữ tự nhiên.

2. **Xử lý ngôn ngữ tự nhiên**: Xử lý ngôn ngữ tự nhiên (NLP) sử dụng các phương pháp và thuật toán để xử lý và phân tích ngôn ngữ tự nhiên. Các phương pháp này bao gồm tách từ, phân tích cú pháp, phân loại ngữ nghĩa, trích xuất thông tin, dịch máy, và nhiều công nghệ khác. NLP kết hợp ngôn ngữ học và khoa học máy tính để xây dựng các công cụ và ứng dụng liên quan đến ngôn ngữ.

3. **Học máy và học sâu**: Học máy chủ yếu dựa trên việc xây dựng và huấn luyện các mô hình máy tính để tự động học từ dữ liệu. Học sâu (deep learning) là một phương pháp học máy dựa trên mạng nơ-ron nhân tạo sâu với nhiều lớp ẩn. Trong NLP, học sâu đã đạt được những tiến bộ đáng kể, đặc biệt là với sự phát triển của mô hình Transformer như BERT, GPT và các biến thể khác, giúp cải thiện hiệu suất của các ứng dụng NLP.

4. **Thống kê và xác suất**: Các phương pháp và khái niệm trong thống kê và xác suất đóng vai trò quan trọng trong NLP. Các mô hình ngôn ngữ dựa trên xác suất như mô hình ngôn ngữ Markov (Markov language model) và mô hình n-gram được sử dụng để ước lượng xác suất của các câu hoặc từ. Các phương pháp thống kê cũng được sử dụng để phân tích dữ liệu ngôn ngữ và đưa ra các kết luận thống kê.

5. **Xử lý dữ liệu lớn**: Một trong những yếu tố quan trọng của NLP hiện đại là khả năng xử lý dữ liệu lớn. Các mô hình NLP phụ

thuộc vào việc huấn luyện trên các tập dữ liệu lớn như corpus văn bản, dữ liệu từ các mạng xã hội và các nguồn dữ liệu khác. Công nghệ xử lý dữ liệu lớn như phân tán, xử lý song song và tính toán đám mây (cloud computing) đóng vai trò quan trọng trong việc xử lý và huấn luyện các mô hình NLP.

Những cơ sở khoa học này cùng với những tiến bộ trong lĩnh vực công nghệ và tính toán đã mang lại những tiến bộ đáng kể trong lĩnh vực NLP và làm cho các ứng dụng NLP trở nên phổ biến và hữu ích trong thực tế.

2.1. Một số khái niệm cơ bản

2.1.1. Ngôn ngữ tự nhiên

Ngôn ngữ tự nhiên (Natural Language) là hình thức giao tiếp và truyền đạt thông tin giữa con người thông qua các phương tiện ngôn ngữ như từ ngữ, ngữ pháp và ngữ cảnh. Nó là hệ thống ngôn ngữ tự nhiên mà con người sử dụng để diễn đạt ý kiến, truyền đạt thông tin, thể hiện cảm xúc và tương tác với nhau. Đặc điểm chung của ngôn ngữ tự nhiên là nó phản ánh cách thức con người diễn đạt ý nghĩa và ý kiến thông qua việc sử dụng ngôn từ, ngữ pháp, ngữ cảnh và ngữ nghĩa.

2.1.2. Nhập nhằng (Xử lý ngôn ngữ tự nhiên)

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. Việc tìm ra các thuật toán hữu hiệu gây khó khăn không ít cho các nhà lập trình.

2.1.3. 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 cơ sở ví dụ.

2.1.4. Xác suất (Probability)

Thực nghiệm và không gian mẫu Không gian mẫu (Sự kiện cơ sở): Ω.
Ví dụ:

- Tung từng đồng xu: $\Omega = \{\text{head}, \text{tail}\}$.
- Bầu cử: $\Omega = \{\text{yes}, \text{no}\}$.
- Tung xúc xắc: $\Omega = \{1, \ldots, 6\}$.
- Xổ số: $(|\Omega| \approx 10^7 \ldots 10^{12})$.
- Số lượng tai nạn giao thông/năm: $\Omega = N$.
- Lỗi chính tả: $\Omega = Z^*$, Z là một bảng chữ cái, Z^* là tập hợp các chuỗi trong bảng chữ cái ($|\Omega| \approx$ kích thước vốn từ vựng).

Sự kiện (Events) Sự kiện A là một tập các mẫu, $A \subseteq \Omega$, tập tất cả A là 2^{Ω}. Ω là sự kiện chắc chắn, $\emptyset$ là sự kiện không xảy ra.
Ví dụ: Tung đồng xu 3 lần, $\Omega = \{\text{HHH}, \text{HHT}, \text{HTH}, \text{HTT}, \text{THH}, \text{THT}, \text{TTH}, TTT\}$. Tính các trường hợp có đúng 2 lần xuất hiện Tail: $A = \{\text{HTT}, \text{THT}, \text{TTH}\}$. Tất cả Head: $A = \{\text{HHH}\}$.

Xác suất (Probability) Thực hiện một thực nghiệm (experiment) nhiều lần: có bao nhiêu lần sự kiện A xảy ra ("count" c_i). Mỗi lần thực nghiệm này gọi là dãy (bộ). Thực hiện các dãy này nhiều lần, ghi nhớ lại con số c_i. Nếu thực sự thử nghiệm thực nghiệm nhiều lần, tỉ số $\frac{c_i}{T_i}$ (T_i là tổng số lần thực nghiệm trong dãy thứ i) dần tới một hằng số chưa biết. Gọi hằng số này là xác suất của A (ký hiệu: $p(A)$).

Ước lượng xác suất Cách tính như sau: Từ một dãy thực nghiệm: $p(A) = \frac{c_1}{T_1}$. Nếu được nhiều dãy thực nghiệm, tính trung bình cộng của $\frac{c_i}{T_i}$.

Kỳ vọng (Expectation) và phương sai (Variance)

- **Kỳ vọng**: Tổng trọng số giá trị của X, hay giá trị trung bình của biến ngẫu nhiên.
- **Phương sai**: Trung bình bình phương của độ lệch (độ lệch của biến X so với trung bình của nó).

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

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

Hình 2: Kỳ vọng và phương sai

3. Quy 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 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 language). 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 3: Một trình biên dịch

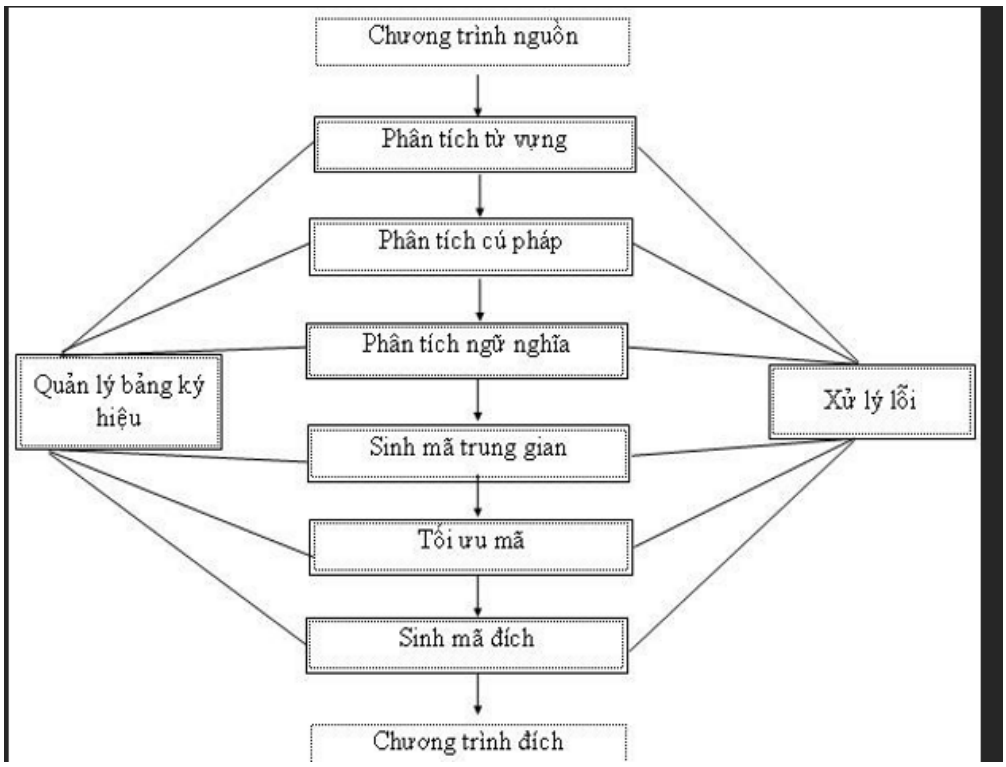
3.1. Các giai đoạn của trình biên dịch

Để dễ hình dung, 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. Một cách phân rã điển hình trình biên dịch được trình bày trong hình sau.

Việc quản lý bảng ký hiệu và xử lý lỗi được thực hiện xuyên suốt qua tất cả các giai đoạn.

3.2. 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).



Hình 4: Các giai đoạn của một trình biên dịch

Ví dụ 1: 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.

3.3. 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 2: Giả sử ngôn ngữ đặc tả bởi các luật sinh sau:

$$\text{Stmt} \rightarrow \text{id} := \text{expr}$$

$$\text{expr} \rightarrow \text{expr} + \text{expr} \mid \text{expr} * \text{expr} \mid \text{id} \mid \text{number}$$

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

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

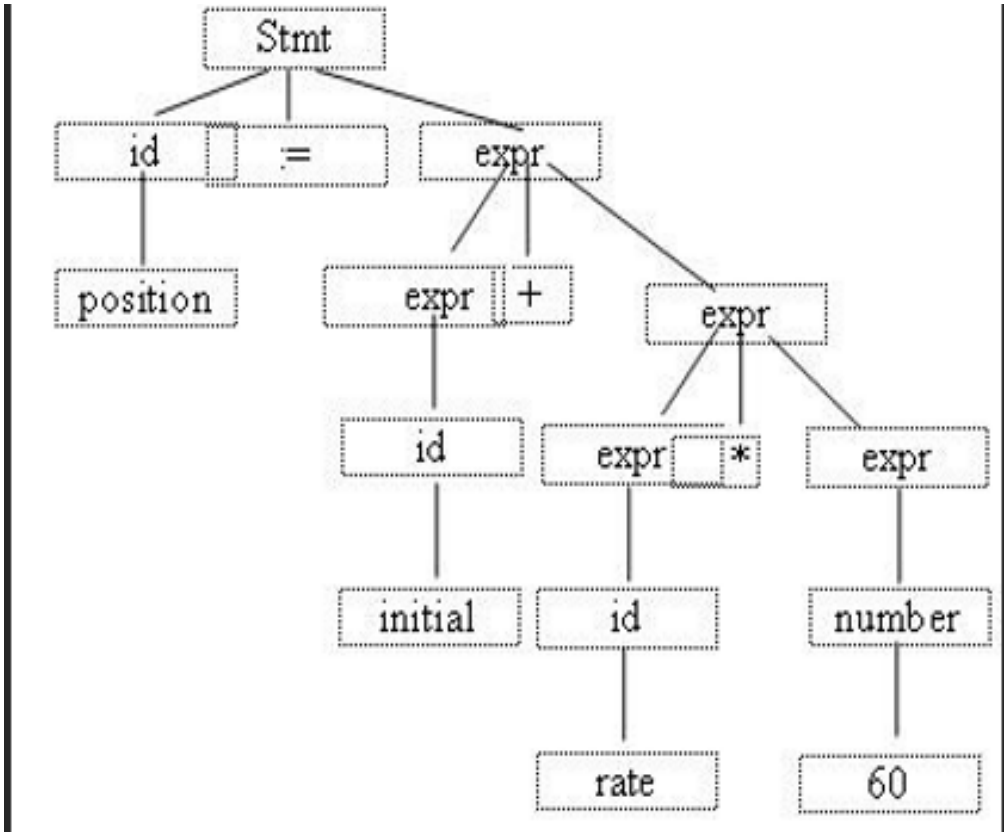
Ví dụ 3:

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)` cũng là những biểu thức.

Câu lệnh (statement) cũng có thể định nghĩa đệ quy:

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`, `if (expr1) then stmt2` đều là các lệnh.

Người ta dùng các quy tắc đệ quy 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 tùy theo công việc thực hiện.



Hình 5: Một cây phân tích cú pháp

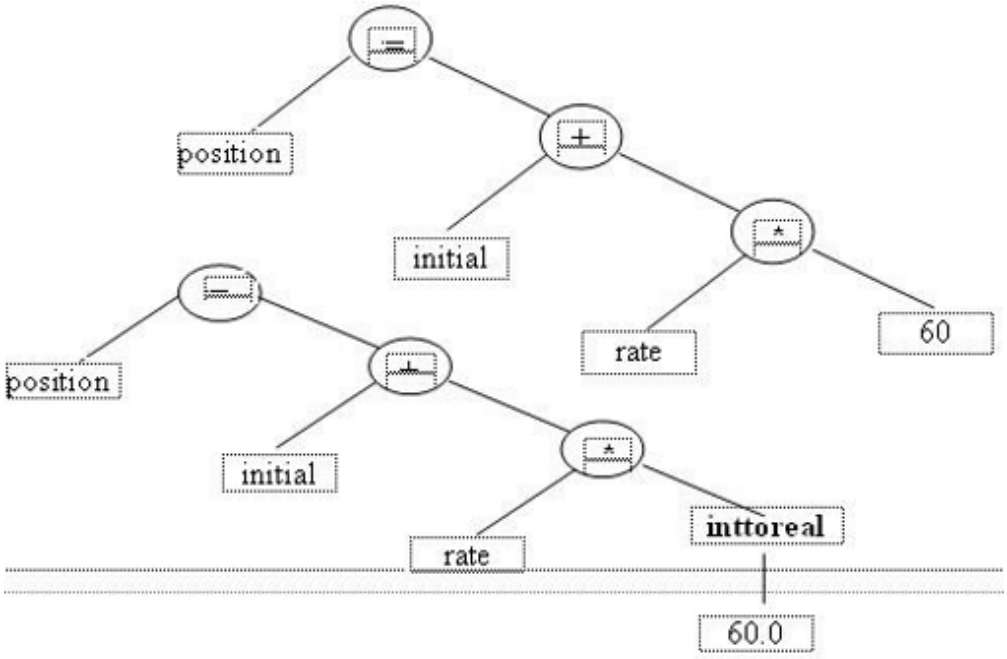
3.4. Phân tích ngữ nghĩa (Semantic Analysis)

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 4: 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 6: Chuyển đổi kiểu trên cây phân tích cú pháp

3.5. Sinh mã trung gian

Sau khi phân tích ngữ nghĩa, một số trình biên dịch sẽ tạo ra một dạng biểu diễn trung gian của chương trình nguồn. Chúng ta có thể xem dạng biểu diễn này như một chương trình dành cho một máy trừu tượng. Chúng có 2 đặc tính quan trọng: dễ sinh và dễ dịch thành chương trình đích.

Dạng biểu diễn trung gian có rất nhiều loại. Thông thường, người ta sử dụng dạng "mã máy 3 địa chỉ" (three-address code), tương tự như dạng hợp ngữ cho một máy mà trong đó mỗi vị trí bộ nhớ có thể đóng vai trò như một thanh ghi.

Mã máy 3 địa chỉ là một dãy các lệnh liên tiếp, mỗi lệnh có thể có tối đa 3 đối số.

Ví dụ 5:

```
t1 := inttoreal(60)
t2 := id3 * t1
t3 := id2 + t2
id1 := t3
```

Dạng trung gian này có một số tính chất:

- Mỗi lệnh chỉ chứa nhiều nhất một toán tử. Do đó khi tạo ra lệnh này, trình biên dịch phải xác định thứ tự các phép toán, ví dụ * thực hiện trước +.
- Trình biên dịch phải tạo ra một biến tạm để lưu trữ giá trị tính toán cho mỗi lệnh.
- Một số lệnh có ít hơn 3 toán hạng.

3.6. Tối ưu mã

Giai đoạn tối ưu mã cố gắng cải thiện mã trung gian để có thể có mã máy thực hiện nhanh hơn. Một số phương pháp tối ưu hóa hoàn toàn bình thường.

Ví dụ 6: Mã trung gian nêu trên có thể tối ưu thành:

```
t1 := id3 * 60.0
id1 := id2 + t1
```

Để tối ưu mã, ta thấy việc đổi số nguyên 60 thành số thực 60.0 có thể thực hiện một lần vào lúc biên dịch, vì vậy có thể loại bỏ phép toán `inttoreal`. Ngoài ra, `t3` chỉ được dùng một lần để chuyển giá trị cho `id1` nên có thể giảm bớt.

Có một khác biệt rất lớn giữa khối lượng tối ưu hóa mã được các trình biên dịch khác nhau thực hiện. Trong những trình biên dịch gọi là "trình biên dịch chuyên tối ưu", một phần thời gian đáng kể được dành cho giai đoạn này. Tuy nhiên, cũng có những phương pháp tối ưu giúp giảm đáng kể thời gian chạy của chương trình nguồn mà không làm chậm đi thời gian dịch quá nhiều.

3.7. Sinh mã đích

Giai đoạn cuối cùng của biên dịch là sinh mã đích, thường là mã máy hoặc mã hợp ngữ. Các vị trí vùng nhớ được chọn lựa cho mỗi biến được chương trình sử dụng. Sau đó, các chỉ thị trung gian được dịch lần lượt thành chuỗi các chỉ thị mã máy. Vấn đề quyết định là việc gán các biến cho các thanh ghi.

Ví dụ 7: Sử dụng các thanh ghi (chẳng hạn `R1`, `R2`) cho việc sinh mã đích như sau:

```
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```

Toán hạng thứ nhất và thứ hai của mỗi chỉ thị tương ứng mô tả đối tượng nguồn và đích. Chữ `F` trong mỗi chỉ thị cho biết chỉ thị đang xử lý các số chấm động (floating-point). Dấu # để xác định số 60.0 xem như một hằng số.

3.8. Quản lý bảng ký hiệu

Một nhiệm vụ quan trọng của trình biên dịch là ghi lại các định danh được sử dụng trong chương trình nguồn và thu thập các thông tin về các thuộc tính khác nhau của mỗi định danh. Những thuộc tính này có thể cung cấp Hacking Overleaf: Tips and Tricks for Efficient Document Editing thông tin về vị trí lưu trữ được cấp phát cho một định danh, kiểu và tầm vực của định danh, và nếu định danh là tên của một thủ tục thì thuộc tính là các thông tin về số lượng và kiểu của các đối số, phương pháp truyền đối số và kiểu trả về của thủ tục nếu có.

Bảng ký hiệu (symbol table) là một cấu trúc dữ liệu mà mỗi phần tử là một mẩu tin dùng để lưu trữ một định danh, bao gồm các trường lưu giữ ký hiệu và các thuộc tính của nó. Cấu trúc này cho phép tìm kiếm, truy xuất danh biểu một cách nhanh chóng.

Trong quá trình phân tích từ vựng, danh biểu được tìm thấy và nó được đưa vào bảng ký hiệu nhưng nói chung các thuộc tính của nó có thể chưa xác định được trong giai đoạn này.

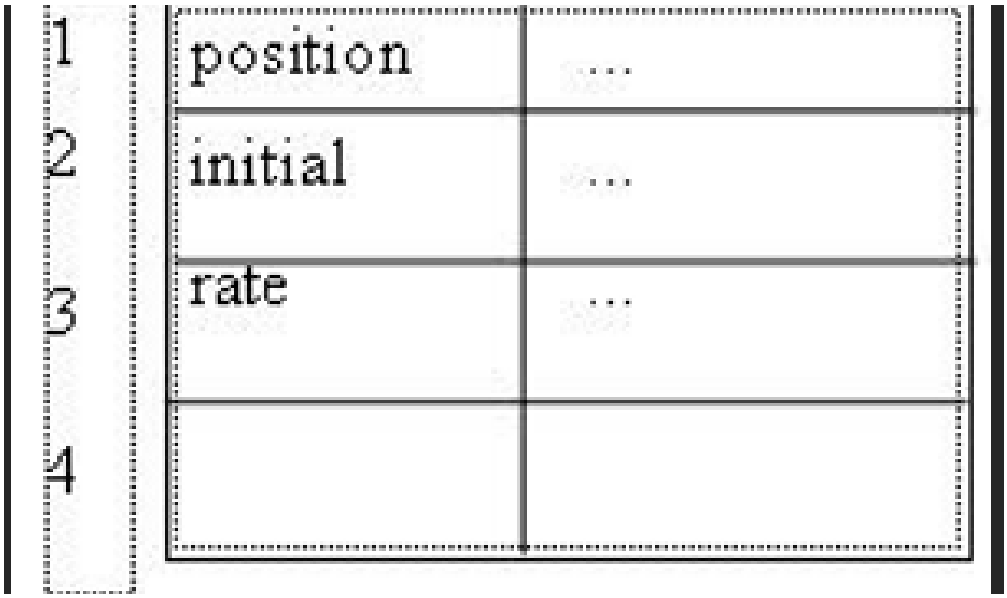
Ví dụ 8: Chẳng hạn, một khai báo trong Pascal có dạng

```
var position, initial, rate : real
```

thì thuộc tính kiểu `real` chưa thể xác định khi các danh biểu được xác định và đưa vào bảng ký hiệu. Các giai đoạn sau đó như phân tích

ngữ nghĩa và sinh mã trung gian mới đưa thêm các thông tin này vào và sử dụng chúng.

Nói chung giai đoạn sinh mã thường đưa các thông tin chi tiết về vị trí lưu trữ dành cho định danh và sẽ sử dụng chúng khi cần thiết.



Hình 7: Chuyển đổi kiểu trên cây phân tích cú pháp

3.9. Xử lý lỗi

Mỗi giai đoạn có thể gặp nhiều lỗi, tuy nhiên sau khi phát hiện ra lỗi, tùy thuộc vào trình biên dịch mà có các cách xử lý lỗi khác nhau, chẳng hạn:

- Dừng và thông báo lỗi khi gặp lỗi đầu tiên (Pascal).
- Ghi nhận lỗi và tiếp tục quá trình dịch (C).

Giai đoạn phân tích từ vựng thường gặp lỗi khi các ký tự không thể ghép thành một token. Giai đoạn phân tích cú pháp gặp lỗi khi các token không thể kết hợp với nhau theo đúng cấu trúc ngôn ngữ. Giai đoạn phân tích ngữ nghĩa báo lỗi khi các toán hạng có kiểu không đúng yêu cầu của phép toán hay các kết cấu không có nghĩa đối với thao tác thực hiện mặc dù chúng hoàn toàn đúng về mặt cú pháp.

3.10. Nhóm 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).

3.10.1. 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.

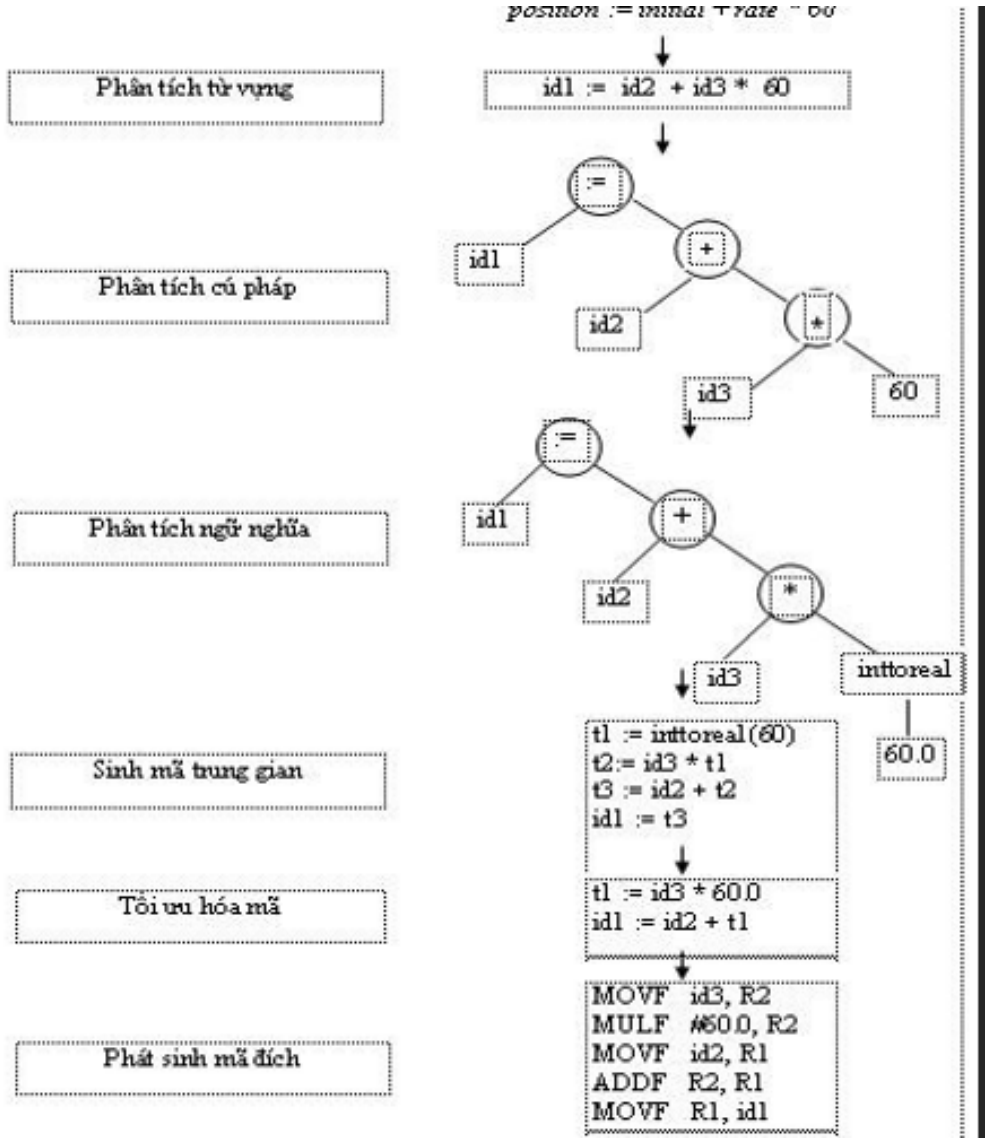
Front end cũng bao gồm cả việc xử lý lỗi xuất hiện trong từng giai đoạn.

3.10.2. 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 hóa mã, phát sinh mã đích cùng với việc xử lý lỗi và các thao tác trên bảng ký hiệu.

4. Các phương pháp phân tích cú pháp

Có nhiều phương pháp phân tích cú pháp được sử dụng trong trình biên dịch. Dưới đây là một số phương pháp phân tích cú pháp phổ biến:



Hình 8: Minh họa các giai đoạn biên dịch một biểu thức

1. **Phân tích theo phương pháp LL (Left-to-Right, Leftmost derivation)**: Phương pháp này dựa trên việc áp dụng ngữ pháp phi ngữ cảnh (context-free grammar) từ trái sang phải, tìm kiếm theo chiều từ trái sang phải trên cấu trúc cây cú pháp. Phân tích LL thường được sử dụng trong các ngôn ngữ lập trình như Pascal và C++.

2. **Phân tích theo phương pháp LR (Left-to-Right, Rightmost derivation)**: Phương pháp này tìm kiếm theo chiều từ trái sang phải trên cấu trúc cây cú pháp. Phân tích LR phổ biến trong các ngôn ngữ lập trình như Java và C# và thường được thực hiện bằng cách sử dụng các trình phân tích cú pháp tự động như Yacc và

Bison.

3. **Phân tích theo phương pháp Recursive Descent**: Phương pháp này dựa trên việc sử dụng các quy tắc phân tích đệ quy để phân tích cú pháp. Mỗi quy tắc ngữ pháp sẽ tương ứng với một hàm phân tích trong mã nguồn của trình biên dịch. Phân tích đệ quy có thể được triển khai một cách đơn giản và dễ hiểu, nhưng có thể gặp vấn đề với các ngôn ngữ có ngữ pháp trái đệ quy.

4. **Phân tích theo phương pháp Shift-Reduce (Dịch-Chuyển)**: Phương pháp này dựa trên nguyên tắc dịch-chuyển (shift) và rút gọn (reduce) để xây dựng cây cú pháp. Nó sử dụng một ngăn xếp (stack) để theo dõi các thành phần đã phân tích và một bảng phân tích (parsing table) để quyết định các hành động dịch-chuyển và rút gọn. Phương pháp Shift-Reduce thường được sử dụng trong các trình biên dịch phân tích cú pháp từ trái sang phải như trình biên dịch LR.

5. **Phân tích theo phương pháp LL(k) và LR(k)**: Đây là các phương pháp được mở rộng của phân tích LL và LR, trong đó k đại diện cho số ký tự được xem trước (lookahead characters). Bằng cách xem trước nhiều ký tự, phân tích LL(k) và LR(k) có thể xử lý được nhiều loại ngữ pháp phức tạp hơn so với LL và LR thông thường.

5. Các ứng dụng của xử lý ngôn ngữ tự nhiên

Các ứng dụng của xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) rất đa dạng và phong phú. Dưới đây là một số ví dụ về các ứng dụng quan trọng của NLP:

- **Hệ thống tìm kiếm thông tin**: NLP được sử dụng trong công cụ tìm kiếm web để hiểu ý định của người dùng và truy xuất các tài liệu phù hợp. Nó giúp cải thiện khả năng tìm kiếm và đưa ra kết quả chính xác hơn.

- **Xử lý ngôn ngữ tự nhiên trong các ứng dụng di động**: NLP được sử dụng trong các ứng dụng di động như trợ lý ảo, bàn phím

dự đoán, ghi chú giọng nói và dịch thuật tự động. Nó giúp tạo ra trải nghiệm người dùng tiện ích và tương tác thông qua ngôn ngữ tự nhiên.

- **Tóm tắt văn bản và trích xuất thông tin**: NLP có thể tạo ra các bản tóm tắt tự động từ các văn bản dài và trích xuất thông tin quan trọng từ các nguồn dữ liệu lớn. Điều này hỗ trợ việc tìm kiếm và tiếp cận thông tin một cách nhanh chóng và hiệu quả.
- **Xử lý ngôn ngữ tự nhiên trong hỗ trợ khách hàng**: NLP được sử dụng trong các chatbot và hệ thống tự động trả lời câu hỏi để cung cấp hỗ trợ khách hàng tự động và nhanh chóng. Nó có thể hiểu và xử lý câu hỏi của người dùng, cung cấp thông tin, giải đáp câu hỏi và giải quyết các vấn đề cơ bản.
- **Dịch máy**: NLP được sử dụng trong công cụ dịch máy để dịch văn bản từ một ngôn ngữ sang ngôn ngữ khác. Công nghệ dịch máy ngày càng tiến bộ và có thể hỗ trợ giao tiếp và trao đổi thông tin giữa các ngôn ngữ khác nhau.
- **Phân loại và phân tích văn bản**: NLP có thể được sử dụng để phân loại và phân tích văn bản trong nhiều ngữ cảnh khác nhau, bao gồm phân loại tin tức, phát hiện thư rác, phân tích ý kiến, nhận dạng thực thể, và phân tích ngôn ngữ tự nhiên y tế.

CHƯƠNG II. GIỚI THIỆU VỀ MÔI TRƯỜNG VÀ CÁC THƯ VIỆN SỬ DỤNG

6. Môi trường thực nghiệm

Google Colab, viết tắt của Google Colaboratory, là một môi trường phát triển và chia sẻ mã nguồn mở dựa trên trình duyệt, được cung cấp bởi Google. Colab cho phép bạn viết và thực thi mã Python trong các trình duyệt web mà không cần cài đặt môi trường phát triển tích hợp (IDE) hoặc thực hiện cài đặt phần mềm trên máy tính cá nhân.

Một số điểm nổi bật về môi trường Google Colab:

- **Máy chủ ảo và tài nguyên mạnh mẽ**: Colab cung cấp máy chủ ảo mạnh mẽ với GPU và TPU để thực thi mã. Điều này rất hữu ích khi làm việc với các tác vụ liên quan đến học máy, xử lý ảnh, xử lý ngôn ngữ tự nhiên và nhiều lĩnh vực khác.

- **Jupyter Notebook**: Colab sử dụng giao diện Jupyter Notebook, cho phép bạn tạo và chia sẻ tệp notebook chứa mã, văn bản, hình ảnh và đồ thị. Giao diện notebook tương tác giúp bạn thực thi từng phần mã một và xem kết quả ngay lập tức.

- **Khả năng chia sẻ**: Bạn có thể chia sẻ tệp notebook Colab với người khác để họ cùng xem và chỉnh sửa. Điều này thuận tiện cho việc làm việc nhóm, hướng dẫn và chia sẻ kiến thức.

- **Tích hợp với Google Drive**: Colab tích hợp tốt với Google Drive, cho phép bạn lưu trữ và truy cập các tệp notebook và dữ liệu trên Google Drive của bạn.

- **Thư viện hỗ trợ**: Colab được cài đặt sẵn với nhiều thư viện phổ biến như `Numpy`, `Pandas`, `Matplotlib` và `TensorFlow`, giúp bạn tiếp cận các công cụ phổ biến trong phân tích dữ liệu và học máy.

- **Kết nối liên tục**: Bạn có thể kết nối Colab với GitHub để quản lý mã nguồn và dễ dàng làm việc với kho lưu trữ mã nguồn.

7. Các thư viện được sử dụng

- **NLTK**: Thư viện NLTK - Natural Language Toolkit là một trong những thư viện open-source xử lý ngôn ngữ tự nhiên. Được viết bằng Python và với ưu điểm là dễ dàng sử dụng nên thư viện này ngày càng trở nên phổ biến và có được một cộng đồng lớn mạnh. Thư viện cung cấp hơn 50 kho dữ liệu văn bản khác nhau (corpora) và nhiều chức năng để xử lý dữ liệu văn bản để phục vụ cho nhiều mục đích khác nhau.

- **re**: Thư viện `re` trong Python là một thư viện chuỗi mẫu (regular expression) được sử dụng để tìm kiếm và xử lý các chuỗi văn bản

theo các quy tắc cụ thể. Nó cung cấp các công cụ mạnh mẽ để thao tác, so khớp và thay thế các chuỗi với mẫu chuỗi phức tạp.

- **sklearn (scikit-learn)**: `scikit-learn`, thường được gọi là `sklearn`, là một thư viện phổ biến trong Python dùng cho machine learning và data mining. Nó cung cấp các công cụ cho việc xây dựng và huấn luyện các mô hình học máy, thực hiện các quy trình tiền xử lý dữ liệu và đánh giá các mô hình.

- **Multiprocessing**: Thư viện `multiprocessing` trong Python cung cấp các công cụ cho việc xử lý đa luồng (parallel processing) và xử lý đa tiến trình (multiprocessing) trong Python. Nó cho phép bạn tận dụng tài nguyên máy tính đa lõi để thực hiện các tác vụ đồng thời và nhanh chóng.

- **Matplotlib.pyplot**: Dùng để tạo và hiển thị đồ thị và biểu đồ. Nó giúp bạn trực quan hóa dữ liệu và tùy chỉnh hình dạng, kiểu đồ thị theo ý muốn.

- **Scipy.io**: Được sử dụng để đọc và ghi dữ liệu từ và vào các tệp tin có định dạng khác nhau. Nó hỗ trợ các chức năng để làm việc với các định dạng dữ liệu như MATLAB (.mat), WAV, NetCDF và nhiều định dạng khác.

- **OS**: Để tương tác với hệ điều hành và thực hiện các hoạt động liên quan đến quản lý tệp tin và thư mục. Nó cung cấp các hàm để thao tác với đường dẫn tập tin, kiểm tra sự tồn tại của tệp tin hoặc thư mục, tạo và xóa thư mục, thay đổi tên tệp tin, và nhiều hoạt động khác liên quan đến hệ thống tệp tin. Đồng thời, thư viện `os` còn cung cấp các hằng số và phương thức để truy cập các biến môi trường hệ thống.

CHƯƠNG III. KẾT QUẢ THỰC NGHIỆM

8. Tổng quan bộ dữ liệu

Tập dữ liệu bao gồm tổng cộng 300 tài liệu. Mỗi tài liệu chứa hai phần chính: một câu mở đầu và một câu chi tiết về vấn đề được đề cập trong câu mở đầu. Các phần này được ngăn cách nhau bằng dấu xuống dòng.

Mặc dù bộ dữ liệu không được gán nhãn, nhưng nó có thể được sử dụng cho bài toán phân cụm để phân loại và nhóm các tài liệu dựa trên sự tương đồng trong nội dung của chúng.

Cách tiếp cận có thể bắt đầu bằng việc tiền xử lý dữ liệu, bao gồm loại bỏ các ký tự đặc biệt không cần thiết, chuyển đổi văn bản về dạng lowercase, loại bỏ stop words (những từ không mang ý nghĩa như "và", "ở", "là",...) và thực hiện tokenization để chia văn bản thành các từ hoặc cụm từ riêng lẻ.

Sau đó, có thể sử dụng các kỹ thuật phân cụm như K-means để phân nhóm các tài liệu dựa trên sự tương đồng trong nội dung của chúng.

Để thực hiện phân cụm hiệu quả, việc lựa chọn các đặc trưng phù hợp từ văn bản cũng rất quan trọng. Có thể sử dụng các phương pháp như TF-IDF (Term Frequency-Inverse Document Frequency) hoặc Word Embeddings (nhúng từ) để biểu diễn văn bản thành các vector số học để áp dụng các thuật toán phân cụm.

Việc thử nghiệm và đánh giá các kỹ thuật phân cụm trên bộ dữ liệu này cũng sẽ cần thiết để hiểu rõ hơn về cách các tài liệu có thể được nhóm lại dựa trên nội dung của chúng.

9. Quy trình thực nghiệm

Sau khi thu thập được bộ dữ liệu, mô hình phân cụm sẽ trải qua các giai đoạn chính sau:

9.1. Giai đoạn 1: Tiền xử lý dữ liệu

Đối với mô hình phân cụm K-Means, dữ liệu đầu vào là dữ liệu kiểu số. Do đó, ta cần chuyển đổi dữ liệu sang dạng số. Đối với dung

lượng dữ liệu còn hạn chế, việc phân biệt chữ viết hoa và viết thường là không cần thiết, vì vậy, ta sẽ đưa tất cả các mẫu về dạng chữ thường. Tiếp theo, thực hiện quá trình phân chia văn bản ra thành các từ riêng lẻ, gọi là các "token", nhằm mục đích chia văn bản thành các phần nhỏ hơn như từ, dấu câu, hay ký tự đặc biệt để dễ dàng xử lý và phân tích ngôn ngữ. Sau đó, thực hiện tạo Bigram ($n = 2$) cho các văn bản, tức là ghép cặp các từ hoặc các đơn vị từ trong văn bản để tạo thành cặp từ liên tiếp. Điều này được tạo ra nhằm mục đích xác định ngữ cảnh của một từ dựa trên từ liền trước, có thể phân tích ngữ cảnh, cũng như tính xác suất của một từ sẽ xuất hiện sau đó. Cuối cùng, thực hiện loại bỏ các từ được gọi là "stop word" như "trời ơi", "như là", "chắc vậy" và các loại dấu câu không cần thiết như "!, #, $, %".

Sau khi làm các bước như trên, tiến hành chuyển đổi dữ liệu từ kiểu chữ sang dạng số thực sử dụng phương pháp TF-IDF nhằm biểu diễn mức độ quan trọng của một từ trong văn bản cũng như toàn bộ tài liệu. Kế đến, ta sử dụng phương pháp TruncatedSVD, nhằm giảm số chiều của dữ liệu đầu vào, giữ lại những đặc trưng quan trọng giúp tăng khả năng cũng như tốc độ tính toán, cải thiện hiệu suất mô hình.

9.2. Giai đoạn 2: Phân cụm dữ liệu

Sau khi đã tiền xử lý, dữ liệu sẽ trở thành dạng mảng có kích thước $(300, 15)$. Trong đó, có 300 mẫu dữ liệu, mỗi mẫu bao gồm 15 đặc trưng là 15 cột. Mỗi mẫu trong bộ dữ liệu là một vector. Từ đây, ta đã hoàn thành việc xử lý đầu vào cho bài toán phân cụm K-Means. Tiếp đó, đưa vào mô hình K-Means với số cụm mong muốn $k = 7$. Do dữ liệu cho việc phân cụm là còn ít, nên việc test sẽ sử dụng dữ liệu tự tạo có liên quan đến một cụm nào đó để xác định cụm.

9.3. Giai đoạn 3: Đánh giá mô hình

Đầu ra của mô hình phân cụm sẽ là cụm tương ứng với từng mẫu dữ liệu, sau đó được đưa đi đánh giá bằng phương pháp tính Silhouette Score, Davies-Bouldin Index và Calinski-Harabasz Index.

10. Các tham số, thư viện và môi trường thực nghiệm

Tham số của mô hình được thiết lập như sau:

Mô hình	Các tham số thiết lập	Giải thích

K-Means Clustering	`_nehbo = 7`
	`ra_ = 42`
	`_s = 400`
		• Số lượng cụm sau khi đánh giá là 7
		• Xác định việc tạo số ngẫu nhiên để khởi tạo tâm. Sử dụng int để xác định tính ngẫu nhiên.
		• Số lần lặp tối đa là 400

Table 1: Tham số mô hình K-Means

Các thư viện được sử dụng:

- **NLTK**: Thư viện NLTK - Natural Language Toolkit là một trong những thư viện open-source xử lý ngôn ngữ tự nhiên. Được viết bằng Python và với ưu điểm là dễ dàng sử dụng nên thư viện này ngày càng trở nên phổ biến và có được một cộng đồng lớn mạnh. Thư viện cung cấp hơn 50 kho dữ liệu văn bản khác nhau (corpora) và nhiều chức năng để xử lý dữ liệu văn bản để phục vụ cho nhiều mục đích khác nhau.

- **sklearn (scikit-learn)**: `scikit-learn`, thường được gọi là `sklearn`, là một thư viện phổ biến trong Python dùng cho machine learning và data mining. Nó cung cấp các công cụ cho việc xây dựng và huấn luyện các mô hình học máy, các hàm tính toán, thực hiện các quy trình tiền xử lý dữ liệu và đánh giá các mô hình.

- **Multiprocessing**: Thư viện `multiprocessing` trong Python cung cấp các công cụ cho việc xử lý đa luồng (parallel processing) và xử lý đa tiến trình (multiprocessing) trong Python. Nó cho phép bạn tận dụng tài nguyên máy tính đa lõi để thực hiện các tác vụ đồng thời và nhanh chóng.

- **Matplotlib.pyplot**: Dùng để tạo và hiển thị đồ thị và biểu đồ. Nó giúp bạn trực quan hóa dữ liệu và tùy chỉnh hình dạng, kiểu đồ thị theo ý muốn.

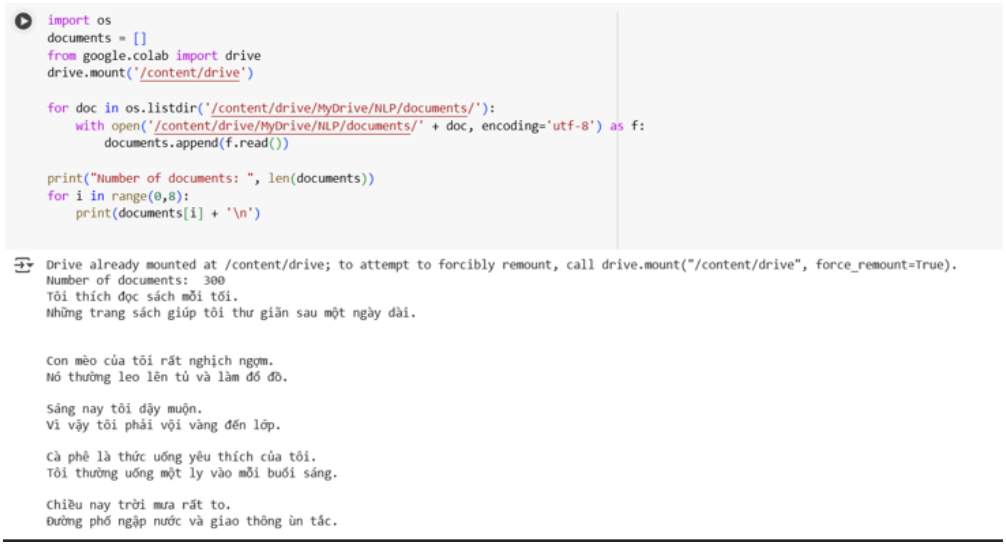
- **os**: Để tương tác với hệ điều hành và thực hiện các hoạt động liên quan đến quản lý tệp tin và thư mục. Nó cung cấp các hàm để thao tác với đường dẫn tập tin, kiểm tra sự tồn tại của tệp tin hoặc thư mục, tạo và xóa thư mục, thay đổi tên tệp tin, và nhiều hoạt động khác liên quan đến hệ thống tệp tin. Đồng thời, thư viện `os` còn cung cấp các hằng số và phương thức để truy cập các biến môi trường hệ thống.

Tất cả quy trình bao gồm đánh giá và huấn luyện đều được triển khai trên môi trường máy ảo sử dụng hệ điều hành Linux 6.1.58+ và phần cứng do Google Colab cung cấp: bao gồm 01 CPU Intel(R) Xeon(R) @ 2.20GHz, 12.7 GB RAM.

11. Các kết quả đạt được

11.1. Kết quả chạy theo giai đoạn thực hiện

- Bước đầu tiên, ta sẽ tiến hành đọc dữ liệu từ file dữ liệu. Điều này được thực hiện bằng cách khai báo các thư viện, truy cập vào thư mục chứa dữ liệu và hiển thị 3 mẫu dữ liệu đầu tiên

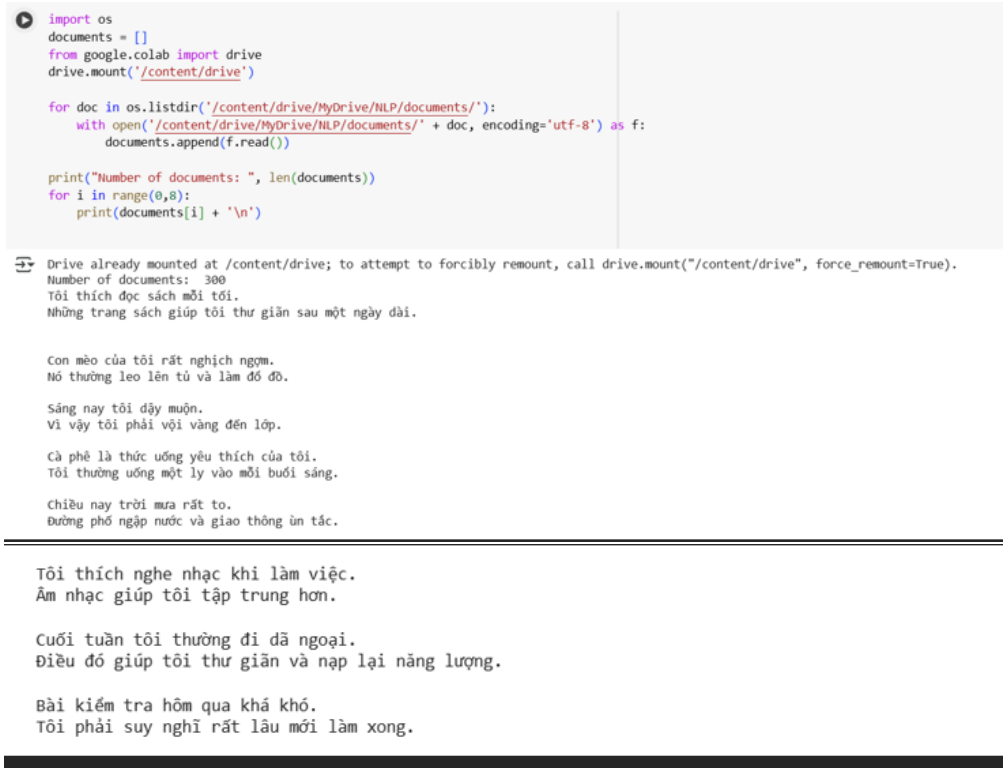


```
import os
documents = []
from google.colab import drive
drive.mount('/content/drive')

for doc in os.listdir('/content/drive/MyDrive/NLP/documents/'):
    with open('/content/drive/MyDrive/NLP/documents/' + doc, encoding='utf-8') as f:
        documents.append(f.read())

print("Number of documents: ", len(documents))
for i in range(0,8):
    print(documents[i] + '\n')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Number of documents:  300
Tôi thích đọc sách mỗi tối.
Những trang sách giúp tôi thư giãn sau một ngày dài.


Con mèo của tôi rất nghịch ngợm.
Nó thường leo lên tủ và làm đổ đồ.

Sáng nay tôi dậy muộn.
Vì vậy tôi phải vội vàng đến lớp.

Cà phê là thức uống yêu thích của tôi.
Tôi thường uống một ly vào mỗi buổi sáng.

Chiều nay trời mưa rất to.
Đường phố ngập nước và giao thông ùn tắc.
```

Hình 9: Tạo danh sách lưu từng mẫu dữ liệu

.



```
import os
documents = []
from google.colab import drive
drive.mount('/content/drive')

for doc in os.listdir('/content/drive/MyDrive/NLP/documents/'):
    with open('/content/drive/MyDrive/NLP/documents/' + doc, encoding='utf-8') as f:
        documents.append(f.read())

print("Number of documents: ", len(documents))
for i in range(0,8):
    print(documents[i] + '\n')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Number of documents:  300
Tôi thích đọc sách mỗi tối.
Những trang sách giúp tôi thư giãn sau một ngày dài.


Con mèo của tôi rất nghịch ngợm.
Nó thường leo lên tủ và làm đổ đồ.

Sáng nay tôi dậy muộn.
Vì vậy tôi phải vội vàng đến lớp.

Cà phê là thức uống yêu thích của tôi.
Tôi thường uống một ly vào mỗi buổi sáng.

Chiều nay trời mưa rất to.
Đường phố ngập nước và giao thông ùn tắc.

Tôi thích nghe nhạc khi làm việc.
Âm nhạc giúp tôi tập trung hơn.

Cuối tuần tôi thường đi dã ngoại.
Điều đó giúp tôi thư giãn và nạp lại năng lượng.

Bài kiểm tra hôm qua khá khó.
Tôi phải suy nghĩ rất lâu mới làm xong.
```

Hình 10: Đọc dữ liệu

- Bắt đầu với tiền xử lý, tạo một danh sách để lưu từng mẫu dữ liệu, mỗi mẫu là 1 phần tử của danh sách.

```python
[ ] class Loader:
        def __init__(self, _data) -> None:
            self.data = _data

        def load_data(self):
            data_str = []  # Chuỗi để lưu trữ nội dung các tệp tin
            for doc in os.listdir(self.data):
                with open(f"{self.data}/{doc}", 'r', encoding='utf-8') as f:
                    data_str.append(f.read().splitlines())  # Thêm nội dung của từng tệp tin vào chuỗi
            return data_str  # Trả về chuỗi đã ghép nối
```

Hình 11: Tạo danh sách lưu từng mẫu dữ liệu

- Thực hiện tiền xử lý với các bước tokenizer, tạo Bigram, chuyển chữ thường và xóa "stopword" cũng như các dấu câu không cần thiết.

```python
stop_words = set(open('/content/drive/MyDrive/NLP/STW/stopwords.txt', encoding='utf-8').read().split('\n')[:-1])
puct_set = set([c for c in "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"])

# data = '/content/drive/MyDrive/NLP/documents'
data = '/content/drive/MyDrive/NLP/documents'

class Preprocessing(Loader):
    def __init__(self, _data, stop_words, puct_set) -> None:
        super().__init__(_data)
        self.data_list = self.load_data()
        self.stop_words = stop_words
        self.puct_set = puct_set

    def tokenizer(self, text):
        return word_tokenize(text)

    def generateBigram(self, text):
        words = text.split()
        if len(words) == 1:
            return ''
        bigrams = [words[i] + '_' + words[i+1] for i in range(0, len(words) - 1)]
        return ' '.join(bigrams)

    def removeRedundant(self, text):
        stop_words_set = self.stop_words
        puct_set = self.puct_set
        words = text.split()
        for i in range(0, len(words)):
            if words[i].count('_') == 0 and (words[i] in stop_words_set | puct_set or words[i].isdigit()):
                words[i] = ''
            else:
                sub_words = words[i].split('_')
                if any(w in stop_words_set | puct_set or w.isdigit() for w in sub_words):
                    words[i] = ''
        words = [w for w in words if w != '']
        return ' '.join(words)

    # Đưa method pre vào đây
    def pre(self):
        prep_data = []
        text = self.data_list
        for i in range(len(text)):
            prep_text = ' '.join(text[i])
            prep_text = ' '.join(self.tokenizer(prep_text))
            prep_text = prep_text.lower()
            prep_text = ' '.join(prep_text.split())
            prep_text = self.generateBigram(prep_text)
            prep_text = self.removeRedundant(prep_text)
            prep_data.append(prep_text)
        return prep_data
```

Hình 12: Tiền xử lý dữ liệu

- Thực hiện tiền xử lý với các bước tokenizer, tạo Bigram, chuyển chữ thường và xóa "stopword" cũng như các dấu câu không cần thiết.

```python
stop_words = set(open('/content/drive/MyDrive/NLP/STW/stopwords.txt', encoding='utf-8').read().split('\n')[:-1])
puct_set = set([c for c in "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"])

# data = '/content/drive/MyDrive/NLP/documents'
data = '/content/drive/MyDrive/NLP/documents'

class Preprocessing(Loader):
    def __init__(self, _data, stop_words, puct_set) -> None:
        super().__init__(_data)
        self.data_list = self.load_data()
        self.stop_words = stop_words
        self.puct_set = puct_set

    def tokenizer(self, text):
        return word_tokenize(text)

    def generateBigram(self, text):
        words = text.split()
        if len(words) == 1:
            return ''
        bigrams = [words[i] + '_' + words[i+1] for i in range(0, len(words) - 1)]
        return ' '.join(bigrams)

    def removeRedundant(self, text):
        stop_words_set = self.stop_words
        puct_set = self.puct_set
        words = text.split()
        for i in range(0, len(words)):
            if words[i].count('_') == 0 and (words[i] in stop_words_set | puct_set or words[i].isdigit()):
                words[i] = ''
            else:
                sub_words = words[i].split('_')
                if any(w in stop_words_set | puct_set or w.isdigit() for w in sub_words):
                    words[i] = ''
        words = [w for w in words if w != '']
        return ' '.join(words)

    # Đưa method pre vào đây
    def pre(self):
        prep_data = []
        text = self.data_list
        for i in range(len(text)):
            prep_text = ' '.join(text[i])
            prep_text = ' '.join(self.tokenizer(prep_text))
            prep_text = prep_text.lower()
            prep_text = ' '.join(prep_text.split())
            prep_text = self.generateBigram(prep_text)
            prep_text = self.removeRedundant(prep_text)
            prep_data.append(prep_text)
        return prep_data
```

Hình 13: Tiền xử lý dữ liệu

- Gọi hàm tiền xử lý và in ra 5 mẫu đầu tiên sau tiền xử lý

```python
prep = Preprocessing(data, stop_words, puct_set)
text = prep.pre()

for i in range(0, 5):
    print(text[i] + '\n')
```

```
đọc_sách trang_sách sách_giúp thư_giãn giãn_sau

nghịch_ngợm đố_đồ

dậy_muộn vội_vàng

cà_phê thức_uống uống_yêu

trời_mưa đường_phố phố_ngập giao_thông thông_ùn ùn_tắc
```

Hình 14: Gọi hàm tiền xử lý và in ra 5 mẫu đầu tiên sau tiền xử lý

- Tiếp theo sử dụng TF-IDF và TruncatedSVD để biến dữ liệu thành dạng số và giảm chiều dữ liệu.

```
vectorizer = TfidfVectorizer(token_pattern = "\S+", min_df = 2)
vectors = vectorizer.fit_transform(text)

print("Tf-idf shape: ZAA" + str(vectors.shape))

svd = TruncatedSVD(n_components=15, n_iter=20, random_state=42)
svd_vectors = svd.fit_transform(vectors)
print(svd_vectors.shape)
print("Document 1's Vector : ")
print(svd_vectors[0])
```

```
Tf-idf shape: ZAA(300, 412)
(300, 15)
Document 1's Vector :
[ 0.00744227  0.22076955  0.08781502  0.16996778  0.13360363 -0.10078315
 -0.02589224 -0.10328502 -0.12302297 -0.01987371  0.15213433 -0.06796409
 -0.0089993  -0.25375978  0.03270524]
```

Hình 15: Biến dữ liệu thành dạng số và giảm chiều dữ liệu

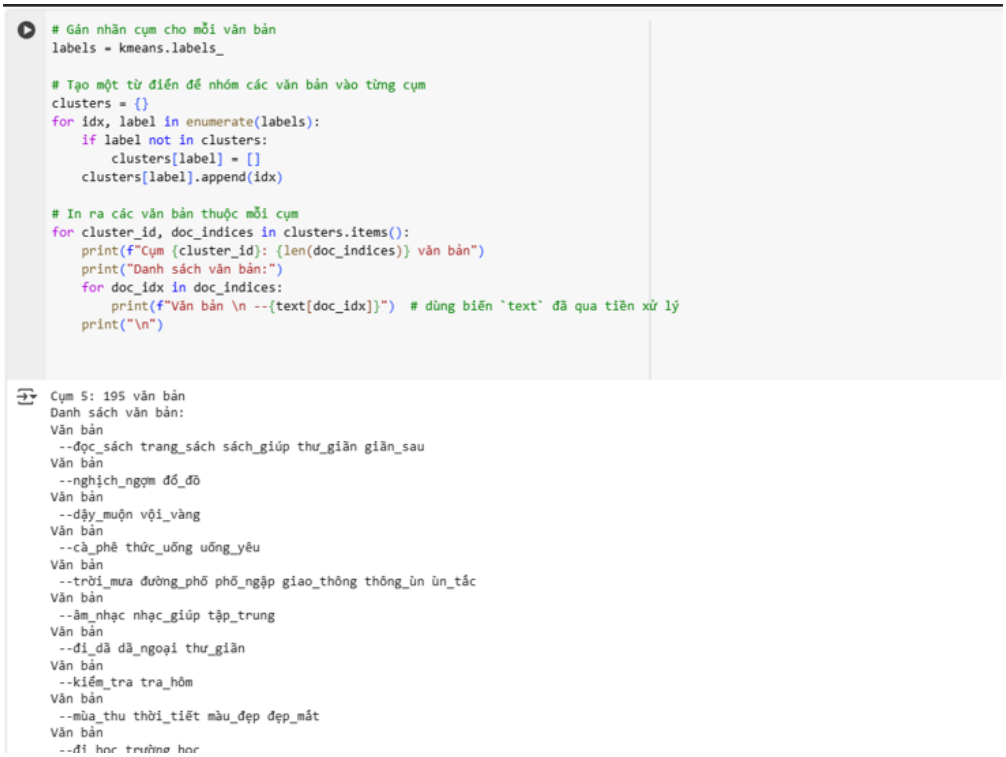
- Sau khi đã xử lý xong dữ liệu đầu vào, ta tiến hành phân cụm dữ liệu với K- Means

```
# Assuming you have svd_vectors, new_data, and kmeans as a trained KMeans model
from sklearn.cluster import KMeans

# Fit KMeans on svd_vectors
n_clusters = 7  # Assuming you used this number of clusters
kmeans = KMeans(n_clusters=n_clusters, random_state=42, max_iter=400)
kmeans.fit(svd_vectors)
```

```
KMeans(max_iter=400, n_clusters=7, random_state=42)
```

Hình 16: P
hân cụm dữ liệu với K-Means

- Hiển thị phân cụm



```
# Gán nhãn cụm cho mỗi văn bản
labels = kmeans.labels_

# Tạo một từ điển để nhóm các văn bản vào từng cụm
clusters = {}
for idx, label in enumerate(labels):
    if label not in clusters:
        clusters[label] = []
    clusters[label].append(idx)

# In ra các văn bản thuộc mỗi cụm
for cluster_id, doc_indices in clusters.items():
    print(f"Cụm {cluster_id}: {len(doc_indices)} văn bản")
    print("Danh sách văn bản:")
    for doc_idx in doc_indices:
        print(f"Văn bản \n --{text[doc_idx]}")  # dùng biến `text` đã qua tiền xử lý
    print("\n")
```

```
Cụm 5: 195 văn bản
Danh sách văn bản:
Văn bản
 --đọc_sách trang_sách sách_giúp thư_giãn giãn_sau
Văn bản
 --nghịch_ngợm đồ_đồ
Văn bản
 --dậy_muộn vội_vàng
Văn bản
 --cà_phê thức_uống uống_yêu
Văn bản
 --trời_mưa đường_phố phố_ngập giao_thông thông_ùn ùn_tắc
Văn bản
 --âm_nhạc nhạc_giúp tập_trung
Văn bản
 --đi_dã dã_ngoại thư_giãn
Văn bản
 --kiểm_tra tra_hôm
Văn bản
 --mùa_thu thời_tiết màu_đẹp đẹp_mắt
Văn bản
 --đi_học trường_học
```

Hình 17: Kết quả phân cụm

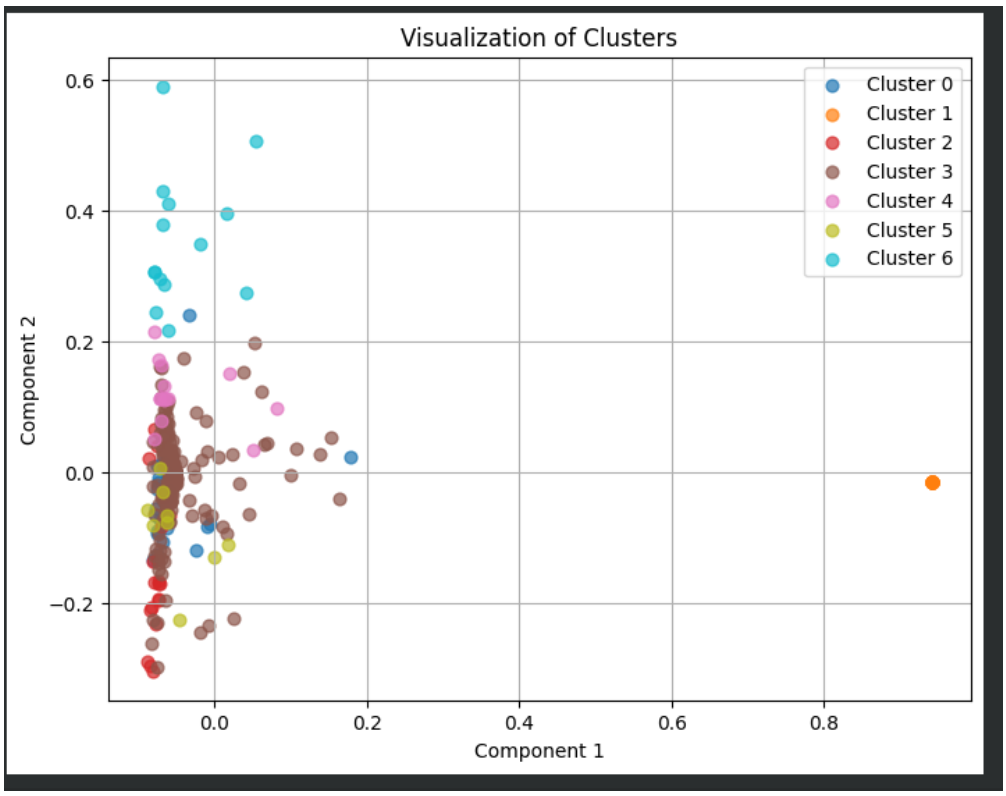
- Cuối cùng, đánh giá mô hình bằng các độ đo Silhouette Score, Davies-Bouldin Index và Calinski-Harabasz Index, chọn 2 đặc trưng để trực quan hóa phân cụm.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
from sklearn.datasets import make_blobs

# Silhouette Score
silhouette_avg = silhouette_score(svd_vectors, labels)
print(f"Silhouette Score: {silhouette_avg}")

# Davies-Bouldin Index
db_index = davies_bouldin_score(svd_vectors, labels)
print(f"Davies-Bouldin Index: {db_index}")

# Calinski-Harabasz Index
ch_index = calinski_harabasz_score(svd_vectors, labels)
print(f"Calinski-Harabasz Index: {ch_index}")
```

```
Silhouette Score: 0.25821018340920737
Davies-Bouldin Index: 1.2627427634085744
Calinski-Harabasz Index: 44.60445679741981
```

Hình 18: Silhouette Score, DaviesBouldin Index và Calinski-Harabasz Index

- Đánh giá mô hình bằng các độ đo Silhouette Score, Davies- Bouldin Index và Calinski-Harabasz Index



Hình 19: Trực quan hóa phân cụm

11.2. Kết quả tổng hợp

Mô hình	Silhouette Score	Davies–Bouldin	Calinski-Haraba
K-Means Clustering	0.258	1.26	44.6

Table 2: Kết quả đánh giá mô hình phân cụm

- **Silhouette Score (Điểm Silhouette)**: Điểm Silhouette đo lường độ tách biệt giữa các cụm đã phân loại. Giá trị của điểm Silhouette nằm trong khoảng $[-1, 1]$. Một giá trị gần 1 cho thấy rằng các điểm trong cùng một cụm gần nhau và xa các cụm khác, trong khi một giá trị gần -1 cho thấy một phân chia sai lầm. Điểm Silhouette càng cao thì phân cụm càng tốt.

- **Davies-Bouldin Index**: Chỉ số Davies-Bouldin đo lường sự tương đồng giữa các cụm. Giá trị càng thấp thể hiện rằng các cụm càng

tách biệt và tốt hơn. Chỉ số này không giới hạn trong một phạm vi cụ thể nhưng giá trị thấp hơn thường tốt hơn.

- **Calinski-Harabasz Index**: Chỉ số Calinski-Harabasz đo lường sự tách biệt giữa các cụm bằng cách tính toán tỉ lệ giữa các phương sai giữa các cụm và giữa các điểm dữ liệu. Giá trị càng cao thì phân cụm càng tốt.

Như vậy, các giá trị đánh giá trong bảng có thể hiểu như sau:

- **Silhouette Score**: 0.258 - Điểm Silhouette khá tốt, cho thấy các điểm trong cụm gần nhau
- **Davies-Bouldin Index**: 1.26 - Chỉ số Davies-Bouldin cũng tương đối thấp, chỉ số này càng gần 0 thì phân cụm càng tốt, và giá trị này cũng cho thấy sự tách biệt tương đối tốt giữa các cụm.
- **Calinski-Harabasz Index**: 44.6 - Chỉ số Calinski-Harabasz cũng khá cao, cho thấy sự tách biệt giữa các cụm.

KẾT LUẬN

Trong bài tập lớn này, chúng ta đã thực hiện phân cụm văn bản để tổ chức và nhóm các tài liệu văn bản dựa trên sự tương đồng của chúng. Phân cụm văn bản là một vấn đề quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên và có nhiều ứng dụng thực tế, như phân loại tài liệu, phân tích chủ đề, và tìm kiếm thông tin.

Thuật toán được sử dụng để tạo ra các cụm (clusters) dựa trên sự tương đồng văn bản với K-Means. Quá trình phân cụm K-Means bao gồm việc khởi tạo các trung tâm cụm ban đầu, phân bổ các văn bản vào các cụm, và cập nhật trung tâm cụm cho đến khi thuật toán hội tụ.

Kết quả phân cụm văn bản cung cấp cho chúng ta cái nhìn tổng quan về cấu trúc và mối quan hệ giữa các văn bản trong tập dữ liệu. Các văn bản thuộc cùng một cụm được cho là có sự tương đồng và chia sẻ các đặc điểm chung. Điều này giúp chúng ta tổ chức dữ liệu,

tìm kiếm thông tin và rút connects trích kiến thức từ các nhóm văn bản tương tự.

Tuy nhiên, việc phân cụm văn bản cũng đặt ra một số thách thức. Các văn bản có thể có sự đa dạng và phức tạp, và việc xác định số lượng cụm phù hợp có thể là một vấn đề khó khăn. Ngoài ra, phương pháp biểu diễn đặc trưng và khoảng cách tính toán cũng có thể ảnh hưởng đến kết quả phân cụm.

Tóm lại, bài tập lớn này đã giúp chúng ta hiểu về quá trình phân cụm văn bản và các phương pháp phân cụm phổ biến. Phân cụm văn bản là một công cụ hữu ích để tổ chức và tìm kiếm thông tin từ các tập dữ liệu văn bản. Tuy nhiên, việc lựa chọn phương pháp và tham số phù hợp là rất quan trọng để đạt được kết quả phân cụm chính xác và ý nghĩa.