ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

KHÓA LUẬN TỐT NGHIỆP

Phát hiện và sửa lỗi văn bản tiếng Việt

Sinh viên thực hiện: Nguyễn Ngọc Tỉnh
Mã SV : 19000297
Lớp: K64A2 - Toán tin
Ngành: Toán tin
(Chương trình đào tạo chuẩn)

HÀ NỘI - 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

KHÓA LUẬN TỐT NGHIỆP

Phát hiện và sửa lỗi văn bản tiếng Việt

Sinh viên thực hiện: Nguyễn Ngọc Tỉnh
Mã SV: 19000297
Lớp: K64A2 - Toán tin
Ngành: Toán tin
(Chương trình đào tạo chuẩn)

Cán bộ hướng dẫn: ThS. Ngô Thế Quyền

HÀ NỘI - 2023

Lời cảm ơn

Đề tài "Phát hiện và sửa lỗi văn bản tiếng Việt" là nội dung em chọn để nghiên cứu và thực hiện khóa luận tốt nghiệp sau thời gian theo học chuyên ngành Toán tin tại trường ĐH Khoa học Tự nhiên, ĐHQGHN. Trong quá trình hoàn thiện khóa luận, em nhận được nhiều sự quan tâm, giúp đỡ từ các thầy cô, bạn bè và gia đình.

Trước hết, em xin gửi lời cảm ơn chân thành đến Thầy/Cô và Hội đồng đã dành thời gian và công sức để đọc và đánh giá khóa luận tốt nghiệp của em.

Em cũng xin gửi lời cảm ơn đặc biệt đến Thầy Ngô Thế Quyền đã tận tâm hướng dẫn và hỗ trợ em trong suốt quá trình nghiên cứu và thực hiện khóa luận. Nhờ những đóng góp và hướng dẫn của Thầy, em đã có cơ hội phát triển khả năng nghiên cứu của mình.

Cuối cùng, em cũng xin cảm ơn bạn bè và gia đình đã đồng hành và ủng hộ em trong quá trình thực hiện khóa luận.

Em xin cam đoan rằng khóa luận tốt nghiệp này là công trình nghiên cứu và thực hiện của riêng mình dưới sự hướng dẫn của Thầy Ngô Thế Quyền. Tất cả các tài liệu tham khảo và nguồn thông tin được trích dẫn rõ ràng và trung thực.

Mặc dù đã cố gắng hoàn thiện khóa luận, em nhận thấy rằng khóa luận của em còn nhiều thiếu sót và hạn chế. Vì vậy, em mong nhận được những ý kiến và nhận xét của các Thầy/Cô để em có thể cải thiện và phát triển đề tài của mình.

Em xin chân thành cảm ơn!

Hà Nội, tháng 05 năm 2023

Nguyễn Ngọc Tỉnh

Mục lục

Danh sách bảng

Danh sách hình vẽ

Mở đầu

Ngôn ngữ là một trong những phương tiện mà nhân loại sử dụng để truyền tải thông tin, ngôn ngữ đóng một vai trò quan trọng trong đời sống xã hội. Trong thời đại số ngày nay, đa số thông tin đều được chia sẻ dưới dạng văn bản như: báo điện tử, các bài viết trên mạng xã hội,... Do vậy, khó thể tránh khỏi việc tồn tại các lỗi chính tả trong văn bản. Những lỗi chính tả này bắt nguồn từ việc gõ nhầm phím hoặc do nhận thức của người soạn thảo.

Trong quá trình truyền tải thông tin, việc có một văn bản chính xác và không chứa lỗi chính tả là rất quan trọng. Lỗi chính tả có thể gây hiểu nhầm và làm mất đi sự đáng tin cậy của người viết. Vì vậy, phát hiện và sửa lỗi chính tả trong văn bản là một vấn đề quan trọng và đang được quan tâm rộng rãi. Nhất là trong thời đại số, với sự phổ biến của văn bản trên Internet và các nền tảng truyền thông xã hội, việc phát hiện và sửa lỗi chính tả trở nên ngày càng cần thiết.

Đề tài khóa luận "Phát hiện và sửa lỗi văn bản tiếng Việt" tập trung vào việc áp dụng mô hình ngôn ngữ N-gram kết hợp với các thuật toán Khoảng cách soạn thảo (*Edit distance*), Chuỗi con chung dài nhất (*Longest Common Subsequence*) để phát hiện và sửa lỗi trong văn bản tiếng Việt. Song song với đó, khóa luận tập trung vào quy trình thu thập dữ liệu huấn luyện và xây dựng tập dữ liệu văn bản lỗi chính tả để đánh giá hiệu suất của mô hình.

Khi thực nghiệm mô hình N-gram đối với tập dữ liệu huấn luyện và dữ liệu kiểm thử, các kết quả ban đầu đạt được khá khả quan. Giai đoạn phát hiện lỗi đạt kết quả 77.22% (độ đo F1-score). Giai đoạn sửa lỗi đạt kết quả 72.96% (độ đo Precision) khi sử dụng Bigram kết hợp thuật toán Khoảng cách soạn thảo và Chuỗi con chung dài nhất.

Khóa luận cũng tìm hiểu và thực nghiệm giai đoạn phát hiện lỗi chính tả theo hướng tiếp cận mô hình học sâu để so sánh độ hiệu quả so với mô hình N-gram. Kết quả thực nghiệm cho thấy mô hình học sâu có hiệu suất vượt trội so với mô hình N-gram. Giai đoạn phát hiện lỗi

sử dụng mô hình học sâu đạt kết quả 97.31% (độ đo F1-score). Từ đó có thể mở rộng thêm nhiều hướng tiếp cận để giải quyết bài toán "Phát hiện và sửa lỗi văn bản tiếng Việt" trong tương lai.

Qua khóa luận này, hy vọng có thể đóng góp vào việc cải thiện chất lượng của các văn bản tiếng Việt thông qua việc phát hiện và sửa các lỗi chính tả.

Khóa luận được tổ chức thành 6 chương:

- **Chương 1**: Mở đầu.
- **Chương 2**: Tổng quan về các lỗi chính tả trong tiếng Việt và trình bày một số nghiên cứu liên quan đến đề tài.
- **Chương 3**: Trình bày cơ sở lý thuyết của các phương pháp được sử dụng để giải quyết bài toán khóa luận nêu ra.
- **Chương 4**: Xây dựng và thực nghiệm mô hình N-gram đối với bài toán phát hiện và sửa lỗi văn bản tiếng Việt.
- **Chương 5**: Giới thiệu mô hình học sâu và thực nghiệm đối với giai đoạn phát hiện lỗi chính tả tiếng Việt.
- **Chương 6**: Trình bày những kết quả đạt được, bên cạnh đó là những hạn chế còn tồn tại của khóa luận và đưa ra định hướng phát triển trong tương lai.

Tổng quan

Ngôn ngữ là một trong những phương tiện mà nhân loại sử dụng để truyền tải thông tin, ngôn ngữ đóng một vai trò quan trọng trong đời sống xã hội. Trong thời đại số ngày nay, đa số thông tin đều được chia sẻ dưới dạng văn bản như: báo điện tử, các bài viết trên mạng xã hội,... Do vậy, khó thể tránh khỏi việc tồn tại các lỗi chính tả trong văn bản. Những lỗi chính tả này bắt nguồn từ việc gõ nhầm phím hoặc do nhận thức của người soạn thảo.

Trong quá trình truyền tải thông tin, việc có một văn bản chính xác và không chứa lỗi chính tả là rất quan trọng. Lỗi chính tả có thể gây hiểu nhầm và làm mất đi sự đáng tin cậy của người viết. Vì vậy, phát hiện và sửa lỗi chính tả trong văn bản là một vấn đề quan trọng và đang được quan tâm rộng rãi. Nhất là trong thời đại số, với sự phổ biến của văn bản trên Internet và các nền tảng truyền thông xã hội, việc phát hiện và sửa lỗi chính tả trở nên ngày càng cần thiết.

2.1 Phát biểu bài toán

Bài toán **"Phát hiện và sửa lỗi văn bản tiếng Việt"** có đầu vào là một văn bản tiếng Việt, mục đích cuối cùng là phát hiện những từ sai chính tả có thể có trong văn bản và đưa ra các giải pháp sửa lỗi chính tả của các từ đó.

Từ bài toán trên, khóa luận tìm hiểu những đề tài liên quan và dựa vào các phương pháp xử lí ngôn ngữ tự nhiên để xây dựng một chương trình phát hiện những lỗi sai chính tả và đưa ra từ gợi ý (hoặc tự động) sửa những lỗi đó.

Do thời gian gấp rút và ngôn ngữ học là một phạm trù rất lớn nên bài toán được giới hạn trong trường hợp sau:

- Lỗi chính tả do đánh máy (lỗi nhập liệu): do các công cụ gõ tiếng Việt như Unikey, Vietkey,... nhập liệu thừa, thiếu hay hoán vị các chữ cái trong một từ.

Ví dụ: "khóa luận" → "khoas luận", "văn bản" → "văn ba3n",...

- Lỗi chính tả do phát âm: lỗi này do đặc điểm phát âm của từng vùng miền, địa phương dẫn đến "nói sao viết vậy".

  Ví dụ: "vô duyên" → "vô diên", "được không"→"được hông",...

- Đối tượng hướng đến của đề tài là văn bản tiếng Việt.

- Chỉ áp dụng đối với trường hợp lỗi chính tả đó là lỗi do vô tình: không tồn tại các lỗi chính tả liên tiếp nhau trong một câu, không tồn tại đồng thời lỗi gõ nhầm và lỗi thừa (thiếu) chữ cái trong một từ.

  Ví dụ: "Nhiều giải pháp, **coong ngệ cũng đc chioa seer** trong cuộc thi về **xu ly vawn ban3**."

- Ngoài ra, một số lỗi về ngữ pháp, cấu trúc câu như lỗi thiếu chủ ngữ, vị ngữ, lặp từ,... cũng được bỏ qua.

  Ví dụ: "Qua tác phẩm này, thể hiện lòng yêu nước của mình." - lỗi thiếu chủ ngữ.

2.2 Các lỗi chính tả thường gặp

Trong văn bản tiếng Việt, các lỗi sai chính tả có thể mắc phải là: sai về âm tiết, sai về ngữ nghĩa hoặc những từ đúng cả về âm tiết và cấu tạo nhưng không tồn tại trong từ điển tiếng Việt.

2.2.1 Lỗi non-word

Lỗi non-word [10] là lỗi mà các từ không có trong từ điển hoặc các từ điển khác như: từ điển tên riêng, từ điển từ viết tắt hay từ điển từ mượn. Nguyên nhân dẫn đến lỗi này có thể là lỗi do đánh máy, lỗi OCR (nhận dạng ký tự quang học), lỗi nhận dạng tiếng nói,... Trong trường hợp hiếm hoi, lỗi phát âm có thể gây ra lỗi non-word, ví dụ: "ngĩa", "zùi", "nghô". Nhưng phần lớn là bắt nguồn từ lỗi đánh máy, ví dụ:

- Kiểu gõ TELEX: “chào” → “chafo”
- Chèn: “chín” → “chính”
- Bớt: “chào” → “chà”
- Thay thế: “mèo” → “mào”
- Hoán vị: “xóa” → “xáo”

	Từ lỗi	**Từ đúng**
Lỗi đánh máy		
Lỗi kiểu gõ TELEX	ow	ơ
Lỗi thiếu dấu	nhieu	nhiều
Lỗi chèn kí tự	xiinh	xinh
Lỗi bớt kí tự	ngỉ	nghỉ
Lỗi thay kí tự	mín	món
Lỗi hoán vị kí tự	xáo	xóa
Lỗi chính tả		
Lỗi ch-/tr-	ngôi chường	ngôi trường
Lỗi x-/-s	sinh đẹp	xinh đẹp
Lỗi l-/n-	lúa lếp	lúa nếp
Lỗi gi-/d-/r-	đơn dản	đơn giản
Lỗi d-/v-	dui dẻ	vui vẻ
Lỗi -c/-t	man mát	man mác
Lỗi -iên/-iêng	thiên liên	thiêng liêng
Lỗi -iu/-ưu	iu ái	ưu ái
Lỗi -n/-ng	cuối cùn	cuối cùng
Lỗi vần ă/â	bế ẫm	bế ẵm
Lỗi dấu hỏi/ngã/sắc	người đỏ	người đó

Bảng 1: Một số lỗi sai chính tả phổ biến [14].

2.2.2 Lỗi real-word

Lỗi real-word [10] là lỗi mà những từ có trong từ điển nhưng được sử dụng sai ngữ cảnh. Nguyên nhân chủ yếu của lỗi này là do cách phát âm của mỗi địa phương, vùng miền (phương ngữ). Đây là một lỗi chính

tả phổ biến bởi tiếng Việt có một số chữ cái, âm tiết có cách phát âm gần giống nhau như: *"d-/r-/gi-", "ch-/tr-", "l-/n-", "s-/x-"*.

Ví dụ: "nhiêu" ↔ "nhiu", "lên" ↔ "nên", "ngã" ↔ "ngả",...

Có thể nhìn một cách tổng quan về các lỗi chính tả thông qua một số ví dụ trong Bảng 1.

2.3 Một số nghiên cứu liên quan

Trong nghiên cứu [10] của mình, tác giả cùng các cộng sự đã sử dụng Bi-gram kết hợp cùng POS (*Parts of Speech*) để tìm ra các âm tiết có khả năng sai chính tả. Với giai đoạn sửa lỗi, nghiên cứu dựa trên Khoảng cách soạn thảo (*Edit distance*), thuật toán SoundEx và một số phương pháp heuristic để đưa ra các ứng viên sửa lỗi.

Các tác giả của bài nghiên cứu [9] đã sử dụng cách tiếp cận dựa trên ngữ cảnh với mô hình N-gram được xây dựng từ dữ liệu văn bản lớn để giải quyết bài toán kiểm tra chính tả.

Một nghiên cứu về kiểm tra chính tả cho các tài liệu văn bản được quét OCR (*Optical Character Recognition*) - nhận dạng ký tự quang học được trình bày tại bài báo [3]. Hướng tiếp cận chính của nghiên cứu này dựa vào các unigram từ dữ liệu văn bản và sự tương đồng gần đúng của các âm tiết.

Tại bài nghiên cứu [1], Aminul Islam và Diana Inkpen đã trình bày phương pháp phát hiện và sửa lỗi chính tả đối với văn bản tiếng Anh sử dụng bộ dữ liệu 3-gram của Google Web 1T cùng thuật toán Chuỗi con chung dài nhất (*Longest Common Subsequence*).

Bài báo [4] trình bày về hệ thống tự động sửa lỗi chính tả. Các tác giả chia hệ thống này làm 3 nhóm: sửa lỗi chính tả tiên nghiệm, sửa lỗi chính tả theo ngữ cảnh và sửa lỗi chính tả bằng mô hình học lỗi (*learning error model*).

2.4 Một số kỹ thuật phát hiện và sửa lỗi chính tả

Một số kỹ thuật được áp dụng để phát hiện và sửa lỗi chính tả trong văn bản được Ritika Mishra và Navjot Kaur giới thiệu trong bài báo [11]:

- Kỹ thuật phát hiện lỗi chính tả:
  - Kỹ thuật tra cứu từ điển: kỹ thuật này được sử dụng để kiểm tra mọi từ của văn bản để nhận biết các từ đó có tồn tại trong từ điển hay không. Nếu có, từ đó là một từ đúng. Ngược lại, nó sẽ được đánh dấu là một từ có thể sai chính tả.
  - Kỹ thuật phân tích N-gram: kỹ thuật này sẽ kiểm tra từng n-gram của chuỗi đầu vào và tra cứu nó trong dữ liệu thống kê n-gram đã được thống kê trước đó, nếu một n-gram không tồn tại hoặc ít xuất hiện thì được coi là lỗi chính tả.
- Kỹ thuật sửa lỗi chính tả:
  - Khoảng cách soạn thảo tối thiểu (*Minimum edit distance*): Khoảng cách soạn thảo tối thiểu là số thao tác (chèn, thay thế, xóa) tối thiểu cần thiết để chuyển một chuỗi này thành một chuỗi khác. Khoảng cách soạn thảo tối thiểu có các thuật toán là Levenshtein, Hamming, LCS (*Longest Common Subsequence*).
  - Khóa tương tự (*Similarity key*): kỹ thuật này ánh xạ mọi chuỗi thành một khóa sao cho các chuỗi được đánh vần giống nhau sẽ có cùng một khóa. Khi một từ được nghi vấn là từ lỗi chính tả, các ký tự của nó được ánh xạ vào các khóa được định sẵn để tất cả các từ khác trong bộ từ vựng có cùng khóa, các từ này được đề xuất là từ để sửa lỗi.
  - Kỹ thuật dựa trên quy tắc (*Rule-based techniques*): kỹ thuật này là các thuật toán biểu diễn kiến thức về các mẫu lỗi chính tả phổ biến dưới dạng các quy tắc để chuyển lỗi chính tả thành

từ hợp lệ. Quá trình tạo ứng viên bao gồm việc áp dụng tất cả các quy tắc áp dụng cho một chuỗi sai chính tả và giữ lại mọi từ trong từ điển hợp lệ cho các kết quả đó.

- Kỹ thuật xác suất (*Probabilistic Techniques*): kỹ thuật này thể hiện xác suất xuất hiện của một từ đã cho sau từ đã cho khác. Các xác suất này có thể được ước tính bằng cách thống kê tần suất n-gram trên một kho dữ liệu lớn.
- Kỹ thuật dựa trên N-gram (*N-gram Based Techniques*): unigram, bigram và trigram được sử dụng theo nhiều cách khác nhau trong kỹ thuật sửa lỗi chính tả. Chúng đã được sử dụng để nắm bắt cú pháp từ vựng của một từ điển và đề xuất các chỉnh sửa phù hợp.
- Kỹ thuật mạng nơ-ron (*Neural Net Techniques*): thuật toán lan truyền ngược (*Back Propagation Algorithm*), Sequence to Sequence,...

2.5 Hướng tiếp cận

Từ các nghiên cứu liên quan, khóa luận tập trung tìm hiểu, nghiên cứu phương pháp phát hiện và sửa lỗi văn bản tiếng Việt dựa trên mô hình ngôn ngữ N-gram.

Cơ sở lý thuyết

3.1 Mô hình ngôn ngữ N-gram

3.1.1 Mô hình ngôn ngữ

Mô hình ngôn ngữ [13] là phân bố xác suất trên các tập văn bản. Cho biết xác suất một câu hay một cụm từ thuộc một ngôn ngữ là bao nhiêu. Mô hình ngôn ngữ được áp dụng vào xử lý ngôn ngữ tự nhiên như: phát hiện lỗi chính tả, dịch máy,...

3.1.2 Tổng quan về mô hình N-gram

N-gram [5] là một chuỗi n phần tử liền nhau của một câu hay một văn bản. Mỗi phần tử là một âm tiết (*syllable*). N-gram được sử dụng với mục đích giúp máy móc phát hiện và hiểu được một âm tiết được sử dụng trong ngữ cảnh nào hay nói cách khác là khiến cho máy móc hiểu được rõ hơn về ngữ nghĩa của âm tiết đó. Tóm lại, mô hình N-gram thể hiện mật độ xuất hiện của một cụm n phần tử liên tiếp nhau.

Có 3 trường hợp Ngram phổ biến là:

- Trường hợp n = 1 (Unigram), tính tần suất xuất hiện của một âm tiết trong câu.
- Trường hợp n = 2 (Bigram) được sử dụng nhiều trong việc phân tích ngữ nghĩa cho các âm tiết.
- Trường hợp n = 3 (Trigram) mục đích được sử dụng tương tự như Bigram.

Ví dụ câu "Tôi là sinh viên" khi được phân tích trong trường hợp Bigram sẽ là:

- Tôi là
- là sinh

- sinh viên

và tương tự với trường hợp Trigram:

- Tôi là sinh
- là sinh viên

Vậy số n-gram có thể có trong một câu sẽ là $x - n + 1$ với x là số âm tiết trong câu. Khi n $>$ 3 thì sẽ có 4-gram, 5-gram,... Với n càng lớn thì độ chính xác của mô hình càng cao tuy nhiên điều này dẫn đến số trường hợp càng lớn khiến cho mô hình càng trở nên phức tạp.

3.1.3 Công thức tính xác suất

Coi S là một chuỗi các từ S = $w_1, w_2, ...w_{m-1}, w_m$ [5].

Khi đó theo công thức xác suất Bayes $P(AB) = P(B|A) \times P(A)$ ta có:

$$\begin{aligned} P(S) &= P(w_1 w_2 w_3 ... w_m) \\ &= P(w_m|w_1 w_2 ... w_{m-1}) \times ... \times P(w_3|w_1 w_2) \times P(w_2|w_1) \times P(w_1) \end{aligned}$$

Nhận thấy rằng, theo công thức này thì mỗi từ sẽ liên quan có điều kiện tới toàn bộ các từ đứng trước nó. Tuy nhiên, trong thực tế, S có thể là một văn bản thì việc áp dụng công thức trên sẽ trở nên rất phức tạp và vô cùng tốn tài nguyên hệ thống. Vì vậy, để giải quyết vấn đề này, chuỗi Markov bậc n được sử dụng, với giả thiết 1 từ chỉ phụ thuộc vào n từ đứng trước nó:

$$P(w_m|w_1 w_2 ... w_{m-1}) = P(w_m|w_{m-n} w_{m-n+1} w_{m-n+2} ... w_{m-1})$$

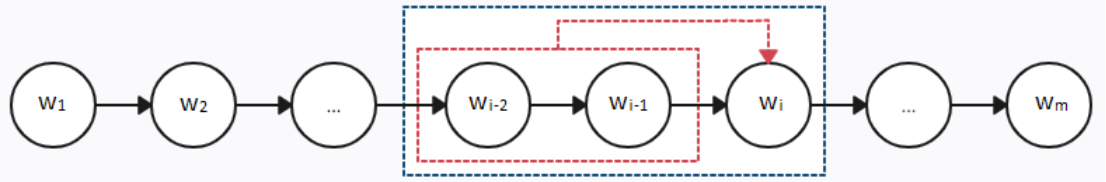
Khi này khả năng xuất hiện của w_i chỉ phụ thuộc vào n từ đã xuất hiện phía trước nó thay vì phải dựa vào toàn bộ các từ phía trước như cách tính sử dụng công thức Bayes. Công thức tính xác suất theo xấp

xỉ Markov sẽ được viết lại như sau:

$$\begin{aligned} P(w_1 w_2 w_3 ... w_m) &= P(w_m | w_{m-n} w_{m-n+1} ... w_{m-1}) \\ &\times P(w_{m-1} | w_{m-n-1} w_{m-n} ... w_{m-2}) \times \cdots \times P(w_3 | w_1 w_2) \\ &\times P(w_2 | w_1) \times P(w_1) \end{aligned}$$

Gọi một mô hình Markov là mô hình Ngram nếu bậc của nó là n-1. Ví dụ, nếu muốn tính xác suất Trigram của một từ w_i thì sẽ xấp xỉ xác suất của mô hình Markov bậc 2, tức dựa trên hai từ trước đó:

$$P(w_1 w_2 ... w_i) = P(w_i | w_{i-2} w_{i-1})$$



Hình 1: Mô hình Markov bậc 2

Tương tự, có thể xấp xỉ xác suất của các mô hình Unigram, Bigram và Trigram như sau:

- Unigram: $P(w_1 w_2 ... w_m) = \prod_i P(w_i)$
- Bigram: $P(w_1 w_2 ... w_m) = \prod_{i=2}^{m} P(w_i | w_{i-1})$,

  với $P(w_i | w_{i-1}) = \frac{count(w_{i-1} w_i)}{count(w_{i-1})}$
- Trigram: $P(w_1 w_2 ... w_m) = \prod_{i=3}^{m} P(w_i | w_{i-2} w_{i-1})$,

  với $P(w_i | w_{i-2} w_{i-1}) = \frac{count(w_{i-2} w_{i-1} w_i)}{count(w_{i-2} w_{i-1})}$

  count(x) là số lần xuất hiện của chuỗi âm tiết x trong dữ liệu huấn luyện mô hình.

3.1.4 Hạn chế của mô hình N-gram

Mô hình N-gram còn tồn tại một số hạn chế. Trước hết là vấn đề mà nhiều mô hình ngôn ngữ gặp phải, đó là khi dữ liệu huấn luyện mô hình càng lớn thì số lượng các cụm n-gram và kích thước của mô hình cũng lớn theo, gây ảnh hưởng không tốt đến mô hình chẳng hạn về thời gian tính toán.

Vấn đề tiếp theo là xây dựng mô hình N-gram dựa trên tập dữ liệu huấn luyện mô hình. Tuy nhiên, khi tính xác suất có thể rơi vào tình huống các cụm n-gram chưa xuất hiện hoặc phân bố không đều, có số lần xuất hiện rất ít trong tập dữ liệu huấn luyện, điều này dẫn đến việc tính toán để đánh giá các câu có chứa các cụm n-gram này không chính xác.

Cụ thể hơn, nếu áp dụng mô hình hoạt động theo công thức xác suất được nêu ở phần trên, tức chỉ đơn thuần là ước lượng dựa trên việc "đếm", hai tình huống sau có thể xảy ra:

- Tử số bằng 0 : tức là count(x) = 0, hay không tồn tại cụm từ x trong dữ liệu huấn luyện.

- Mẫu số bằng 0: count(y) = 0 khi không tồn tại cụm từ y trong tập dữ liệu, công thức phạm vào lỗi chia cho số không, vì vậy xác suất không thể được ước tính.

Với trường hợp tập dữ liệu huấn luyện không thể phủ được hết các trường hợp trong thực tế, một giá trị xác suất không xác định có thể gây ảnh hưởng lớn đến kết quả của mô hình.

3.1.5 Phương pháp làm mịn

Phương pháp làm mịn được đưa ra để cải thiện một phần các hạn chế được nêu ra ở phần trên. Từ đó tính được chính xác hơn khả năng xuất hiện của các cụm n-gram.

Trước hết, có thể hiểu sơ qua về phương pháp làm mịn[5] như sau:

- Gán một giá trị khác 0 (đủ lớn) cho các cụm n-gram không có mặt trong tập dữ liệu huấn luyện.
- Gán lại một giá trị phù hợp cho các cụm n-gram đã xuất hiện trong tập dữ liệu huấn luyện ban đầu để thỏa mãn tổng xác suất của tất cả các cụm n-gram khác nhau phải bằng 1.

Có một số phương pháp làm mịn là: Discounting, Back-off, Interpolation. Phương pháp được sử dụng trong khóa luận là phương pháp chiết khấu (Discounting) với thuật toán Add-one (Laplace). Thuật toán này sẽ làm mịn bằng cách cộng thêm 1 vào số lần xuất hiện của mỗi cụm n-gram trước khi tính xác suất. Tất cả các cụm n-gram mà có số lần xuất hiện bằng 0 thì sau khi cộng thêm 1 sẽ là 1, số lần xuất hiện là 1 thì sẽ được tăng lên là 2 và tương tự như vậy. Khả năng xuất hiện của mỗi n-gram sẽ được tính bởi công thức:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{count(w_{i-n+1}...w_{i-1}w_i)+1}{count(w_{i-n+1}...w_{i-1})+V}$$

trong đó V là kích thước bộ từ vựng.

Tuy nhiên, với công thức trên, mô hình sẽ cho ra các kết quả không mấy tốt đẹp nếu kích thước bộ từ vựng lớn trong khi số lần xuất hiện của 1 cụm n-gram nào đó là rất ít. Để cải thiện thêm tính hiệu quả của mô hình, có thể sử dụng công thức sau:

$$P(w_1w_2...w_n) = \frac{count(w_1w_2...w_n)+\lambda}{count(w_1w_2...w_{n-1})+\lambda.M}$$

trong đó M là số cụm (n-1)-gram có thể có và $\lambda \in [0, 1]$:

- $\lambda = 0$: xấp xỉ Markov.
- $\lambda = 1$: add-one.

3.2 Khoảng cách soạn thảo

3.2.1 Giới thiệu

Khoảng cách soạn thảo, hay được biết đến với tên "Khoảng cách Levenshtein". Khoảng cách soạn thảo tính toán số lượng tối thiểu các

phép biến đổi chữ cái cần thiết để chuyển từ này thành từ khác. Sự khác biệt giữa các chuỗi càng lớn thì khoảng cách soạn thảo nhận được càng lớn.

Trong khóa luận, thuật toán "Khoảng cách soạn thảo" được ứng dụng để đưa ra các gợi ý sửa lỗi chính tả trong chương trình.

3.2.2 Định nghĩa

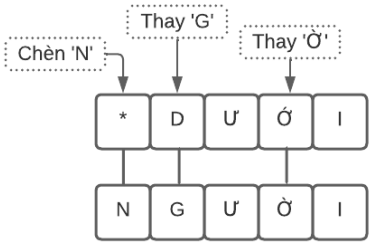
Coi 2 từ là 2 chuỗi kí tự A = $a_1a_2a_3...a_m$ và B = $b_1b_2b_3...b_n$, $|A| = m$, $|B| = n$. Khoảng cách soạn thảo là số phép biến đổi để chuyển chuỗi A thành chuỗi B hoặc ngược lại với các phép biến đổi sau:

- Chèn
- Xóa
- Thay thế

Mỗi lần thực hiện biến đổi chỉ được thực hiện với một kí tự trong chuỗi. Ví dụ, để thực hiện việc chuyển đổi từ A = "họ" sang B = "học", cần thực hiện việc chèn một kí tự 'c' vào chuỗi A và lúc này, chuỗi A đã được chuyển thành chuỗi B. Do chuyển đổi từ chuỗi A sang B chỉ mất một thao tác là chèn kí tự nên khoảng cách soạn thảo giữa chuỗi A và chuỗi B là 1.

Xét thêm một ví dụ, chuỗi A = "dưới" và B = "người". Khi chuyển đổi từ chuỗi A sang chuỗi B cần lần lượt thực hiện các thao tác:

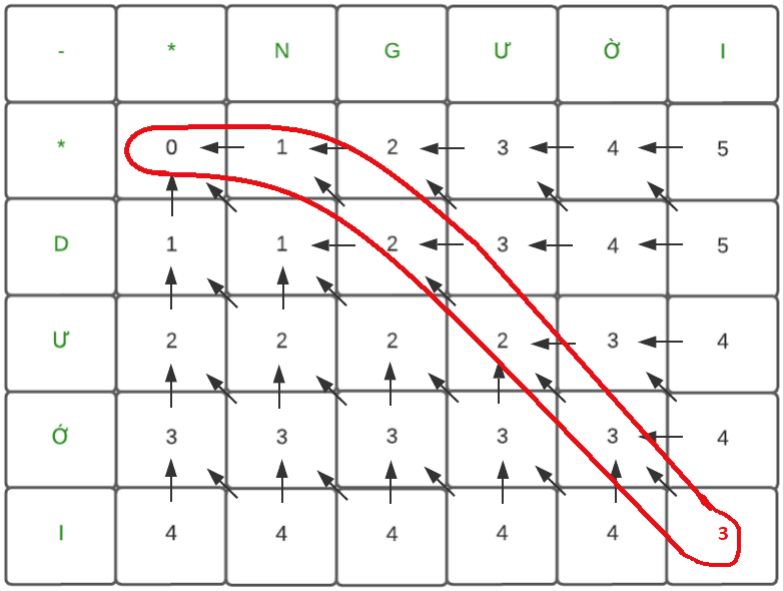
"dưới" $\longrightarrow$ "dười" $\longrightarrow$ "gười" $\longrightarrow$ "người"



Hình 2: Edit_distance("dưới", "người")

Như vậy, để chuyển từ "dưới" $\longrightarrow$ "người" cần tối thiểu 3 bước thực hiện. Vì vậy khoảng cách soạn thảo của "dưới" và "người" là 3.

3.2.3 Thuật toán



Hình 3: Ma trận tính khoảng cách soạn thảo.

Để tính toán Khoảng cách soạn thảo nhỏ nhất, thuật toán quy hoạch động sẽ được sử dụng, tính toán trên mảng 2 chiều (m+1)×(n+1), với m, n là độ dài của chuỗi cần tính [6].

Mã giả của thuật toán [6]:

Bước 1: Khởi tạo.

T(i, 0) = i

T(0, j) = j

Bước 2: Lặp.

for i in $\overline{1, ..., m}$ **do**:

  for j in $\overline{1, ..., n}$ **do**:

    $T(i, j) = min(del, ins, sub)$

Bước 3: Trả về khoảng cách soạn thảo nhỏ nhất.

min_edit_distance = T(m,n)

Trong đó, del, ins, sub được định nghĩa như sau:
del = T(i-1, j) + 1 (up)
ins = T(i, j-1) + 1 (left)
sub = T(i-1, j-1) + c (diag),

$$c = \begin{cases} 1 & \text{nếu } A(i) \neq B(j) \\ 0 & \text{nếu } A(i) = B(j) \end{cases}$$

Thuật toán tìm khoảng cách soạn thảo nhỏ nhất được mô tả như sau:

- Mỗi ô T(i,j) của mảng thể hiện khoảng cách của i kí tự đầu tiên của chuỗi A và j kí tự đầu tiên của chuỗi B.
  - Ví dụ, khi 'i' = 0, khoảng cách soạn thảo giữa chuỗi A có độ dài bằng 0 và chuỗi B có độ dài bằng j là j khoảng cách.
  - Tương tự, trường hợp 'j' = 0, khoảng cách soạn thảo giữa chuỗi B có độ dài bằng 0 và chuỗi A có độ dài bằng i là i khoảng cách.
- Trong quá trình thao tác trên ma trận này, ô T(i,j) thể hiện khoảng cách giữa chuỗi A có độ dài i, chuỗi B có độ dài j.
  - Nếu kí tự thứ i của chuỗi A và kí tự thứ j của chuỗi B giống nhau thì điền vào ô T(i,j) giá trị của ô T(i-1,j-1) - ô này thể hiện khoảng cách sửa đổi của i-1 kí tự đầu tiên của chuỗi A và j-1 kí tự đầu tiên của chuỗi B.
  - Nếu kí tự thứ i của chuỗi A khác kí tự thứ j của chuỗi B thì chọn ra giá trị nhỏ nhất trong 3 trường hợp sau (phép nào cho khoảng cách ngắn nhất thì sẽ được chọn):
    * Xóa kí tự thứ i của chuỗi A và tính khoảng cách sửa đổi giữa kí tự thứ i-1 của chuỗi A và kí tự thứ j của chuỗi B. del = T(i-1, j) + 1, T(i-1,j) là khoảng cách sửa đổi giữa m-1 kí tự của chuỗi A và n kí tự của chuỗi B, 1 là chi phí

của thao tác xóa.

* Chèn: ins = T(i, j-1) + 1, chèn kí tự cuối cùng của chuỗi B vào chuỗi A.
* Thay thế: sub = T(i-1, j-1) + c, thay thế kí tự cuối cùng của chuỗi B vào chuỗi A, c = 1 nếu kí tự A(i) khác B(j), c = 0 nếu ngược lại.

3.3 Chuỗi con chung dài nhất

3.3.1 Giới thiệu

Chuỗi con chung dài nhất (*Longest Common Subsequence*) [2] là chuỗi con có độ dài lớn nhất mà xuất hiện trong cả hai chuỗi đầu vào. Đối với hai chuỗi A và B, chuỗi con chung dài nhất là một chuỗi mới được tạo ra bằng cách loại bỏ các kí tự không cần thiết và giữ lại các kí tự giống nhau trong cả hai chuỗi, theo đúng thứ tự xuất hiện của chúng. Trong quy hoạch động, chuỗi con chung dài nhất được tính bằng cách sử dụng bảng để lưu trữ các giá trị tương ứng.

Ví dụ, cho hai chuỗi A = "ABCBDAB" và B = "BDCAB". Chuỗi con chung dài nhất của hai chuỗi này là "BCAB" hoặc "BDAB" với độ dài bằng 4.

Chuỗi con chung dài nhất không nhất thiết phải là một chuỗi con liên tiếp trong cả hai chuỗi đầu vào. Nó có thể có các kí tự được xếp theo thứ tự khác nhau, nhưng vẫn duy trì thứ tự tương đối giữa các kí tự giống nhau.

Chuỗi con chung dài nhất là một khái niệm quan trọng trong nhiều bài toán, bao gồm tìm kiếm sự tương đồng giữa các chuỗi, sửa lỗi chính tả và nhiều ứng dụng khác trong xử lý ngôn ngữ và xử lý dữ liệu chuỗi.

3.3.2 Bài toán

Cho 2 chuỗi kí tự A = $a_1a_2a_3...a_m$ và B = $b_1b_2b_3...b_n$, $|A| = m$, $|B| = n$. Tìm chuỗi con chung dài nhất của A và B.

Theo [12], bài toán tìm chuỗi con chung dài nhất có cấu trúc con tối ưu, tức bài toán có thể được chia làm các bài toán nhỏ hơn cho đến khi lời giải trở nên tầm thường. Chuỗi con chung dài nhất nói riêng có các bài toán con chồng chéo: các bài toán con cấp cao thường sử dụng lại các giải pháp của các bài toán con cấp thấp hơn. Các vấn đề với hai tính chất này có thể tuân theo phương pháp quy hoạch động, tức các giải pháp của các bài toán con được lưu lại để sử dụng lại khi gặp lại bài toán con đó.

Hàm tìm chuỗi con chung dài nhất được định nghĩa như sau:

$$\text{LCS}(\text{A}_i, B_j) = \begin{cases} \epsilon & ;\ i = 0 \text{ hoặc } j = 0 \\ LCS(A_{i-1}, B_{j-1}).concat(a_i) & ;\ i, j > 0 \text{ và } a_i = b_j \\ max\{LCS(A_i, B_{j-1}), LCS(A_{i-1}, B_j)\} & ;\ i, j > 0 \text{ và } a_i \neq b_j \end{cases}$$

$\text{LCS}(\text{A}_i, B_j)$ thể hiện chuỗi con chung dài nhất của 2 chuỗi A và B. Cần so sánh a_i, b_j để tìm chuỗi con chung dài nhất của A_i, B_j. Nếu chúng giống nhau, $LCS(A_i, B_j)$ được nối với phần tử a_i. Nếu chúng khác nhau thì chuỗi dài nhất trong hai chuỗi $LCS(A_i, B_{j-1})$, $LCS(A_{i-1}, B_j)$ sẽ được chọn. Trường hợp A_i hoặc B_j là rỗng thì chuỗi con chung dài nhất là chuỗi rỗng.

Dựa vào hàm tìm chuỗi con chung dài nhất, chiều dài của một chuỗi con chung dài nhất được tính theo công thức :

$$\text{c[i, j]} = \begin{cases} 0 & ;\ i = 0 \text{ hoặc } j = 0 \\ c[i-1, j-1] + 1 & ;\ i, j > 0 \text{ và } a_i = b_j \\ max\{c[i, j-1], c[i-1, j]\} & ;\ i, j > 0 \text{ và } a_i \neq b_j \end{cases}$$

3.3.3 Thuật toán

Để tính độ dài của chuỗi con chung dài nhất của chuỗi A[1...m] và B[1...n], thuật toán tính toán trên mảng 2 chiều $C_{(m+1)\times(n+1)}$. C[m+1, n+1] là độ dài của chuỗi con chung dài nhất của A và B [12].

Mã giả:

```
C = array(0...m, 0...n)
for i in 1, ..., m do:
    C[i, 0] = 0
for j in 1, ..., n do:
    C[0, j] = 0
for i in 1, ..., m do
    for i in 1, ..., n do
        if A[i] = B[j] then
            C[i,j] = C[i-1,j-1] + 1
        else
            C[i,j] = max(C[i,j-1], C[i-1,j])
return C[m, n]
```

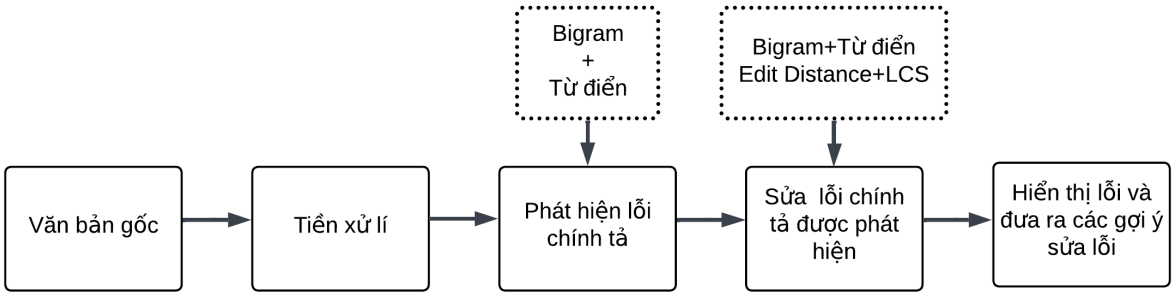
		G	C	C	C	T	A	G	C	G
	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1
C	0	1	2	2	2	2	2	2	2	2
G	0	1	2	2	2	2	2	3	3	3
C	0	1	2	3	3	3	3	3	4	4
A	0	1	2	3	3	3	4	4	4	4
A	0	1	2	3	3	3	4	4	4	4
T	0	1	2	3	3	4	4	4	4	4
G	0	1	2	3	3	4	4	5	5	5

Hình 4: Ma trận tính độ dài của chuỗi con chung dài nhất[7].

Mô hình N-gram trong bài toán phát hiện và sửa lỗi văn bản tiếng Việt

4.1 Xây dựng mô hình

4.1.1 Mô hình chung

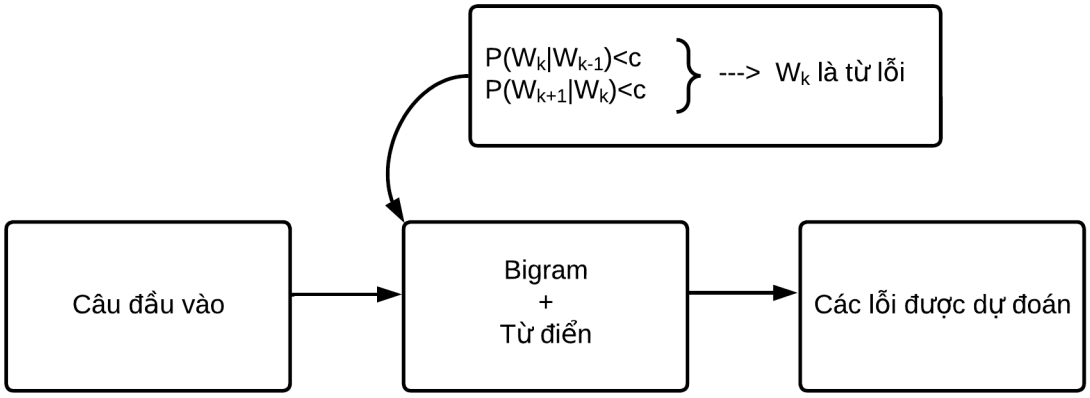


Hình 5: Mô hình phát hiện và sửa lỗi văn bản.

Trước khi đưa vào giai đoạn phát hiện lỗi chính tả, văn bản đầu vào sẽ được tiền xử lý: các chuỗi viết hoa (viết hoa ký tự đầu tiên), các chuỗi có kí tự đầu tiên là số,... được đánh dấu - mặc định coi là đúng chính tả. Ở giai đoạn phát hiện lỗi chính tả, dựa trên Bigram để phát hiện lỗi chính tả trong văn bản đã xử lý trước đó. Với các lỗi được phát hiện, tiếp tục sử dụng Bigram kết hợp thuật toán Khoảng cách soạn thảo cùng thuật toán Chuỗi con chung dài nhất để đưa ra các gợi ý sửa lỗi.

4.1.2 Mô hình phát hiện lỗi

Giai đoạn phát hiện lỗi với Bigram được mô tả ngắn gọn như sau: Với một câu đầu vào S = $W_1W_2W_3...W_n$. Đầu ra sau khi được xử lý sẽ là các từ cùng nhãn kèm theo. Từ đúng kèm theo nhãn 'c', từ có khả năng là sai kèm theo nhãn 'i'.



Hình 6: Mô hình phát hiện lỗi văn bản.

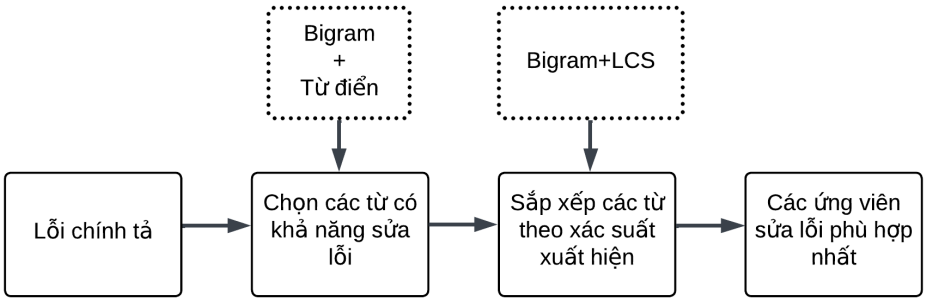
- Xét một cụm từ W = $W_{k-1}W_kW_{k+1}$, giả sử W_{k-1} và W_{k+1} là từ đúng.
- Tính giá trị xác suất của P($W_k|W_{k-1}$) và P($W_k|W_{k+1}$) dựa trên các Bigram đã thống kê.
  - P($W_k|W_{k-1}$): xác suất xuất hiện W_k khi tồn tại W_{k-1}.
  - P($W_{k+1}|W_k$): xác suất xuất hiện của W_{k+1} khi tồn tại W_k.
- Nếu P($W_k|W_{k-1}$) < c và P($W_{k+1}|W_k$) < c thì khả năng W_k sẽ là từ sai chính tả, với c là một "ngưỡng" được chọn.

Để việc chọn "ngưỡng" thuận lợi hơn, khóa luận sử dụng hàm logarit Nepe ln() để biểu diễn lại các giá trị xác suất, điều này giúp giảm độ lệch giữa các giá trị xác suất. Khi sử dụng hàm ln() các giá trị xác suất ban đầu nằm trong khoảng [0,1] sẽ được ánh xạ vào khoảng từ [-∞, 0], tức giá trị xác suất càng nhỏ thì giá trị ln() của xác suất càng nhỏ. Ví dụ, giá trị xác suất P = 0.00003 thì ln(P) = -10.414. Khi đó, việc đặt "ngưỡng" sẽ được biến đổi lại như sau:

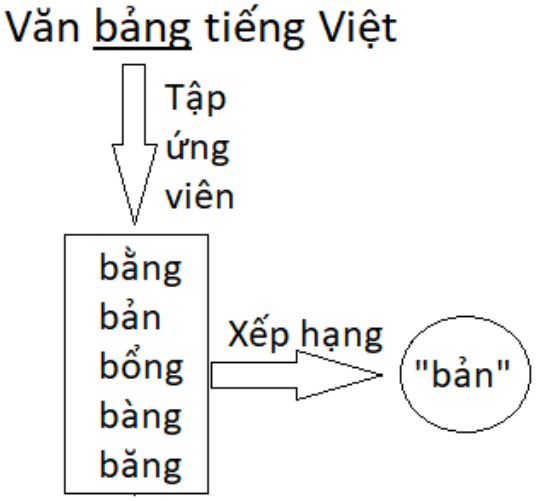
- W_k có thể là từ sai chính tả nếu $-ln(P(W_k|W_{k-1})) > threshold$ và $-ln(P(W_{k+1}|W_k)) > threshold$, $threshold \in (0, +\infty)$.

4.1.3 Mô hình sửa lỗi

Đầu vào của giai đoạn sửa lỗi là kết quả đầu ra của giai đoạn phát hiện lỗi. Ở giai đoạn này, các từ được gán nhãn 'i' sẽ được thay thế bằng các từ gợi ý. Những từ nằm trong bộ từ điển từ vựng có tính liên quan nhất với từ sai chính tả sẽ được chọn làm các từ gợi ý sửa lỗi.



Hình 7: Mô hình sửa lỗi văn bản.



Hình 8: Minh họa đưa ra gợi ý sửa lỗi.

Danh sách các từ được chọn làm phương án sửa lỗi được tạo ra như sau:

- **Xác định các từ gợi ý:** Tính khoảng cách soạn thảo nhỏ nhất của từ sai chính tả với các từ đơn trong từ điển tiếng Việt. Những từ nào thỏa mãn khoảng cách nhỏ nhất sẽ được đưa vào danh sách những từ khả dụng.

- **Tính xác suất xuất hiện:** Tính xác suất xuất hiện của mỗi từ ở danh sách trên dựa vào từ ở vị trí trước hoặc từ ở vị trí sau từ sai chính tả trong câu theo các Bigram đã thống kê.

- **Sắp xếp các từ gợi ý:** Sắp xếp lại các từ theo xác suất xuất hiện kết hợp thuật toán Chuỗi con chung dài nhất.

4.2 Dữ liệu

4.2.1 Thu thập dữ liệu

Điều đầu tiên cần thực hiện là thu thập dữ liệu. Ở giai đoạn này, cần viết một đoạn chương trình ngắn để có thể thu thập dữ liệu văn bản từ các website tiếng Việt. Để dữ liệu được "sạch" và có ít lỗi sai chính tả nhất có thể thì dữ liệu sẽ được thu thập từ các trang báo điện tử uy tín như Báo VnExpress (http://vnexpress.net/), Báo Tuổi trẻ (http://tuoitre.vn/), Báo Thanh niên (http://thanhnien.vn/),...
Chương trình thu thập dữ liệu được xây dựng như sau:

- Cài đặt và sử dụng thư viện **newspaper**, đây là một thư viện Python3.
  (https://pypi.org/project/newspaper3k/)

- Tổng hợp lại đường dẫn (*url*) của các trang báo để trích xuất nội dung văn bản, đây chính là đầu vào của chương trình.

- Sau khi quét các đường dẫn, nội dung của các bài viết sẽ được lưu lại trong một tệp có định dạng *.txt*.

4.2.2 Xử lý dữ liệu

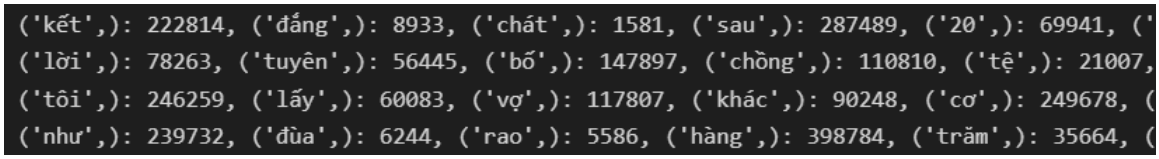
Đây là một bước quan trọng, nhờ đó sẽ "làm sạch" được dữ liệu gốc (dữ liệu được thu thập từ các trang báo) và thuận lợi hơn cho việc thống kê n-gram. Cụ thể, việc này được thực hiện như sau:

- Tách các văn bản thành các câu. Việc tách nhỏ các câu này dựa vào dấu câu và các kí tự đặc biệt như ".:,!?()[]" (sử dụng thư viện NLTK-Natural Language Toolkit).

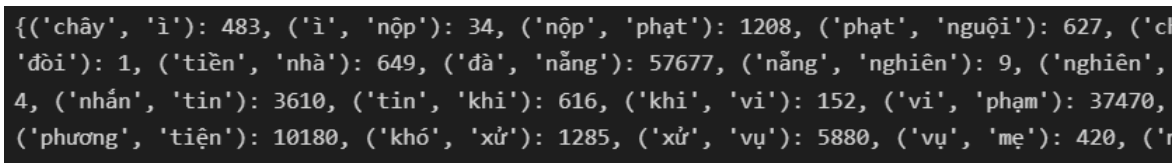
- Sau khi văn bản được tách thành các câu, các câu này tiếp tục được tách thành các token, mỗi token sẽ được lưu trên một dòng (mỗi câu được cách nhau bởi một dòng trống) của tệp có định dạng *.tsv*.

4.2.3 Thống kê n-gram

Ngữ liệu sau khi xử lý được phân tích thành unigram và bigram, sau đó chúng được thống kê tần suất xuất hiện và được lưu trong một danh sách tìm kiếm.



Hình 9: Một số unigram được thống kê.



Hình 10: Một số bigram được thống kê.

4.2.4 Tạo dữ liệu sai chính tả

Để có dữ liệu phục vụ việc kiểm tra và đánh giá mô hình, khóa luận tiếp tục thu thập dữ liệu từ các trang báo điện tử từ đó tạo ra dữ liệu sai chính tả. Để tỉ lệ lỗi sai chính tả được tạo ra một cách hợp lý, khóa luận đã khảo sát bộ dữ liệu từ nguồn `https://github.com/spraakbanken/multiged-2023` . Đây là bộ dữ liệu của dự án Shared task

on Multilingual Grammatical Error Detection (MultiGED-2023) nhằm nghiên cứu và phát triển các mô hình hoặc hệ thống có khả năng phát hiện lỗi ngữ pháp đa ngôn ngữ, cụ thể là: tiếng Czech, tiếng Anh, tiếng Đức, tiếng Ý và tiếng Thụy Điển. Bộ dữ liệu này đã được thu thập, kiểm tra và xử lý bởi các chuyên gia ngôn ngữ vì vậy có thể đảm bảo về độ tin cậy.

Các tỉ lệ lỗi của bộ dữ liệu này được thống kê trong các Bảng 2, 3, 4, 5, 6, 7, 8, 9.

Bảng 2: Tỉ lệ lỗi trong văn bản tiếng Czech.

Czech		
Nguồn	cs_geccc_train	cs_geccc_dev
Tổng số câu	29795	2780
Tổng số token	332866	31939
Token TB trong câu	11.17	11.48
Số token chứa lỗi	67523	8039
Tỉ lệ lỗi	20.29%	25.17%
Đầu câu	8.26%	8.87%
Gần đầu câu	8.92%	10.04%
Giữa câu	69.63%	68.23%
Gần cuối câu	12.54%	11.17%
Cuối câu	0.66%	1.69%

Bảng 3: Tỉ lệ lỗi trong văn bản tiếng Anh.

English		
Nguồn	en_fce_train	en_fce_dev
Tổng số câu	28357	2191
Tổng số token	454730	34748
tiếp tục ở trang tiếp theo...		

Bảng 3 – tiếp tục từ trang trước.

Nguồn	en_fce_train	en_fce_dev
Token TB trong câu	16.03	15.85
Số token chứa lỗi	42899	3460
Tỉ lệ lỗi	9.43%	9.96%
Đầu câu	3.87%	3.7%
Gần đầu câu	4.6%	4.86%
Giữa câu	83.41%	83.32%
Gần cuối câu	7.18%	7.11%
Cuối câu	0.94%	1.01%

Bảng 4: Tỉ lệ lỗi trong văn bản tiếng Anh(tiếp).

English	
Nguồn	en_realec_dev
Tổng số câu	4067
Tổng số token	88008
Token TB trong câu	21.63
Số token chứa lỗi	8103
Tỉ lệ lỗi	9.21%
Đầu câu	3.5%
Gần đầu câu	3.48%
Giữa câu	86.86%
Gần cuối câu	5.29%
Cuối câu	0.86%

Bảng 5: Tỉ lệ lỗi trong văn bản tiếng Đức.

German		
Nguồn	de_falko-merlin_train	de_falko-merlin_dev
Tổng số câu	19239	2503
Tổng số token	305108	39444
Token TB trong câu	15.85	15.75
Số token chứa lỗi	46220	6063
Tỉ lệ lỗi	15.15%	15.37%
Đầu câu	4.18%	4.02%
Gần đầu câu	5.6%	6.04%
Giữa câu	81.36%	81.18%
Gần cuối câu	7.07%	6.83%
Cuối câu	1.79%	1.93%

Bảng 6: Tỉ lệ lỗi trong văn bản tiếng Ý.

Italian		
Nguồn	it_merlin_train	it_merlin_dev
Tổng số câu	6394	758
Tổng số token	80335	9144
Token TB trong câu	12.56	12.06
Số token chứa lỗi	12189	1211
Tỉ lệ lỗi	15.17%	13.24%
Đầu câu	7.74%	7.93%
Gần đầu câu	7.78%	7.68%
Giữa câu	75.95%	76.14%
Gần cuối câu	7.57%	7.35%
Cuối câu	0.96%	0.91%

Bảng 7: Tỉ lệ lỗi trong văn bản tiếng Thụy Điển.

Swedish		
Nguồn	sv_swell_train	sv_swell_dev
Tổng số câu	6729	911
Tổng số token	115203	15685
Token TB trong câu	17.12	17.21
Số token chứa lỗi	21615	2970
Tỉ lệ lỗi	18.76%	18.94%
Đầu câu	4.19%	4.04%
Gần đầu câu	4.74%	5.22%
Giữa câu	82.8%	82.19%
Gần cuối câu	6.69%	7.21%
Cuối câu	1.59%	1.35%

Bảng 8: Tỉ lệ lỗi trong các văn bản trên.

Tổng hợp	
Nguồn	
Tổng số câu	103724
Tổng số token	1507210
Token TB trong câu	14.53
Số token chứa lỗi	220292
Tỉ lệ lỗi	14.62%
Đầu câu	5.72%
Gần đầu câu	6.55%
Giữa câu	77.74%
Gần cuối câu	8.84%
Cuối câu	1.15%

Bảng 9: Số lỗi trung bình trong mỗi câu của các văn bản.

	Số lỗi trung bình trong 1 câu		
Nguồn	<12 token	[12 ,17] token	>17 token
cs_geccc_train	1,39	2,84	4,99
cs_geccc_dev	1,88	3,56	5,61
en_fce_train	0,52	1,38	2,60
en_fce_dev	0,56	1,45	2,79
en_realec_dev	0,86	1,37	2,48
de_falko-merlin_train	1,2	2,14	3,88
de_falko-merlin_dev	1,15	2,14	3,96
it_merlin_train	1,10	2,25	3,72
it_merlin_dev	1,01	1,91	3,15
sv_swell_train	1,34	2,69	5,13
sv_swell_dev	1,36	2,87	5,18
All	1,13	2,15	3,61

Thông qua các số liệu này, có thể tạo dữ liệu sai chính tả dựa theo các tỉ lệ:

- Tỉ lệ lỗi của toàn bộ dữ liệu kiểm thử.
- Tỉ lệ vị trí lỗi trong từng câu.
- Số lỗi trong mỗi câu.

Bên cạnh đó, các từ sai chính tả sẽ được sinh dựa vào các quy tắc tại Bảng 10, Bảng 11 và Bảng 12.

Bảng 10: Luật biến đổi một từ đúng chính tả thành từ sai chính tả.

STT	**Lỗi**	**Biến đổi**
1	Các cặp từ thường gây nhầm lẫn trong tiếng Việt, chúng sẽ được thay thế cho nhau.	sương/xương, sĩ/sỹ, sẽ/sẻ, sã/sả, chỉnh/chỉn, sữa/sửa, chẳng/chẵng, cổ/cỗ, sát/xát, truyện/chuyện, giả/dả, đỡ/đở, giữ/dữ, giã/dã, ngành/ngành, nghề/ngề, xuất/suất, sáng/xáng, sông/xông, xuống/suống, thăm/tham, bạc/bạt, quan/quang, dặn/vặn, chặn/chặng, chắp/chấp, mùi/muồi, chính/chín, sót/xót, chuẩn/chẩn, dành/giành, diếm/giếm, xẻn/sẻn, ấp/ắp, xá/sá, thuyết/thiết, xúc/súc, hàng/hằng, khắc/khắt, khẳng/khảng, mạng/mạn, nhận/nhậm, sát/sáp, suông/suôn, tri/trí, chung/trung, xự/sự, sở/xở.
2	Phụ âm đầu	s/x, gi/d, d/r, ch/tr, ng/n, nh/n, ngh/ng, n/l, qu/w, ph/f, qu/q
3	Phụ âm cuối	ục/uộc, ẩn/uẩn, ùi/uồi, ac/at, ăc/ăt, iêc/iêt, uc/ut, uôc/uôt, ưc/ưt, ươc/ươt, an/ang, ăn/ăng, en/eng, iên/iêng, oan/oang, oăn/oăng, un/ung, uôn/uông, ưn/ưng, ươn/ương, ic/ich, êch/êt, ich/it, ên/ênh, in/inh, ai/ay, iu/ưu, iu/iêu, om/ôm/ơm, ưi/ươi
4	Vần	ă/â, ă/a, o/ô, e/ê

Bảng 11: Luật biến đổi một từ đúng chính tả thành từ sai chính tả (tiếp).

STT	**Lỗi**	**Biến đổi**
5	Lỗi TELEX, biến đổi từ có dấu thành từ tương ứng trường hợp gõ sai kiểu TELEX	“à/af”, “á/as”, “ạ/aj”, “ả/ar”, “ã/ax”, “â/aa”, “ầ/aaf”, “ấ/aas”, “ậ/aaj”, “ẩ/aar”, “ẫ/aax”, “ă/aw”, “ằ/awf”, “ắ/aws”, “ặ/awj”, “ẳ/awr”, “ẵ/awx”, “è/ef”, “é/es”, “ẹ/ej”, “ẻ/er”, “ẽ/ex”, “ê/ee”, “ề/ee”, “ế/ees”, “ệ/eej”, “ể/eer”, “ễ/eex”, “ì/if”, “í/is”, “ị/ij”, “ỉ/ir”, “ĩ/ix”, “ò/of”, “ó/os”, “ọ/oj”, “ỏ/or”, “õ/ox”, “ô/oo”, “ồ/oof”, “ố/oos”, “ộ/ooj”, “ổ/oor”, “ỗ/oox”, “ơ/ow”, “ờ/owf”, “ớ/ows”, “ợ/owj”, “ở/owr”, “ỡ/owx”,
6	Lỗi VNI	‘à/a2’, ‘á/a1’, ‘ạ/a5’, ‘ả/a3’, ‘ã/a4’, ‘â/a6’, ‘ầ/â2’, ‘ấ/â1’, ‘ậ/â5’, ‘ẩ/â3’, ‘ẫ/â4’, ‘ă/a8’, ‘ằ/ă2’, ‘ắ/ă1’, ‘ặ/ă5’, ‘ẳ/ă3’, ‘ẵ/ă4’, ‘è/e2’, ‘é/e1’, ‘ẹ/e5’, ‘ẻ/e3’, ‘ẽ/e4’, ‘ê/e6’, ‘ề/ê2’, ‘ế/ê1’, ‘ệ/ê5’, ‘ể/ê3’, ‘ễ/ê4’, ‘ì/i2’, ‘í/i1’, ‘ị/i5’, ‘ỉ/i3’, ‘ĩ/i4’, ‘ò/o2’, ‘ó/o1’, ‘ọ/o5’, ‘ỏ/o3’, ‘õ/o4’, ‘ô/o6’, ‘ồ/ô2’, ‘ố/ô1’, ‘ộ/ô5’, ‘ổ/ô3’, ‘ỗ/ô4’, ‘ơ/o7’, ‘ờ/ơ2’, ‘ớ/ơ1’, ‘ợ/ơ5’, ‘ở/ơ3’, ‘ỡ/ơ4’, ‘ù/u2’, ‘ú/u1’, ‘ụ/u5’, ‘ủ/u3’, ‘ũ/u4’, ‘ư/u7’, ‘ừ/ư2’, ‘ứ/ư1’, ‘ự/ư5’, ‘ử/ư3’, ‘ữ/ư4’, ‘ỳ/y2’, ‘ý/y1’, ‘ỵ/y5’, ‘ỷ/y3’, ‘ỹ/y4’, ‘đ/d9’

Bảng 12: Luật biến đổi một từ đúng chính tả thành từ sai chính tả (tiếp).

STT	Lỗi	Biến đổi
7	Thanh điệu - hỏi/ngã, ngã/nặng	ả/ã/ạ, ẩ/ẫ/ậ, ẳ/ẵ/ặ, ổ/ỗ/ộ, ở/ỡ/ợ, ủ/ũ/ụ, ể/ễ/ệ, ẻ/ẽ/ẹ, ỉ/ĩ/ị, ỷ/ỹ/ỵ, ủ/ũ/ụ, ử/ữ/ự
8	Teencode	mình/mk|mik|mjk, vô/zô|zo|vo, vậy/zậy|zay|za, phải/fải|fai, biết/bit|biet, rồi/rùi|ròi|r, bây/bi|bay, giờ/h, đi/di|dj, gì/j, em/e, được/dc|đc, tao|tôi/t, chồng/ck, vợ/vk
9	Từ không dấu	‘à|á|ạ|ả|ã|â|ầ|ấ|ậ|ẩ|ẫ|ă|ằ|ắ|ặ|ẳ|ẵ/a’, ‘è|é|ẹ|ẻ|ẽ|ê|ề|ế|ệ|ể|ễ/e’, ‘ì|í|ị|ỉ|ĩ/i’, ‘ò|ó|ọ|ỏ|õ|ô|ồ|ố|ộ|ổ|ỗ|ơ|ờ|ớ|ợ|ở|ỡ/o’, ‘ù|ú|ụ|ủ|ũ|ư|ừ|ứ|ự|ử|ữ/u’, ‘ỳ|ý|ỵ|ỷ|ỹ/y’, ‘đ/d’

Bảng 13 thể hiện thông số của tập dữ liệu sai chính tả tiếng Việt được tạo ra.

Bảng 13: Thông số dữ liệu sai chính tả được sử dụng đánh giá mô hình.

Tổng số câu	36644
Tổng số token	933983
Số token chứa lỗi	82896
Lỗi real-word	50384
Lỗi non-word	32512
Tỉ lệ lỗi	8.88%
Đầu câu	3.7%
Gần đầu câu	4.24%
Giữa câu	84.25%
Gần cuối câu	3.98%
tiếp tục ở trang tiếp theo...	

Bảng 13 – tiếp tục từ trang trước.

Cuối câu	3.83%
Số lỗi trong câu $<$ 12 token	0.88
Số lỗi trong câu [12, 17] token	1.42
Số lỗi trong câu $>$ 17 token	2.85

4.3 Thực nghiệm

Quá trình thực nghiệm:

- Chuẩn bị ngữ liệu: Ngữ liệu dùng để thống kê các unigram và bigram có kích thước khoảng 1GB dạng văn bản (khoảng hơn 11 triệu câu).

- Tạo từ điển tiếng Việt: Từ điển được tạo ra dựa vào tập dữ liệu từ điển tiếng Việt của tác giả Hồ Ngọc Đức. Từ điển bao gồm gần 74000 các từ và cụm từ trong tiếng Việt.

  (Trang nguồn của bộ dữ liệu từ điển: `http://www.informatik.uni-leipzig.de/~duc/software/misc/wordlist.html`)

- Thống kê unigram và bigram: Sử dụng ngữ liệu đã được chuẩn bị để thống kê các unigram và bigram. Số lượng unigram và bigram được thống kê từ ngữ liệu lần lượt là 269002 và 5139553.

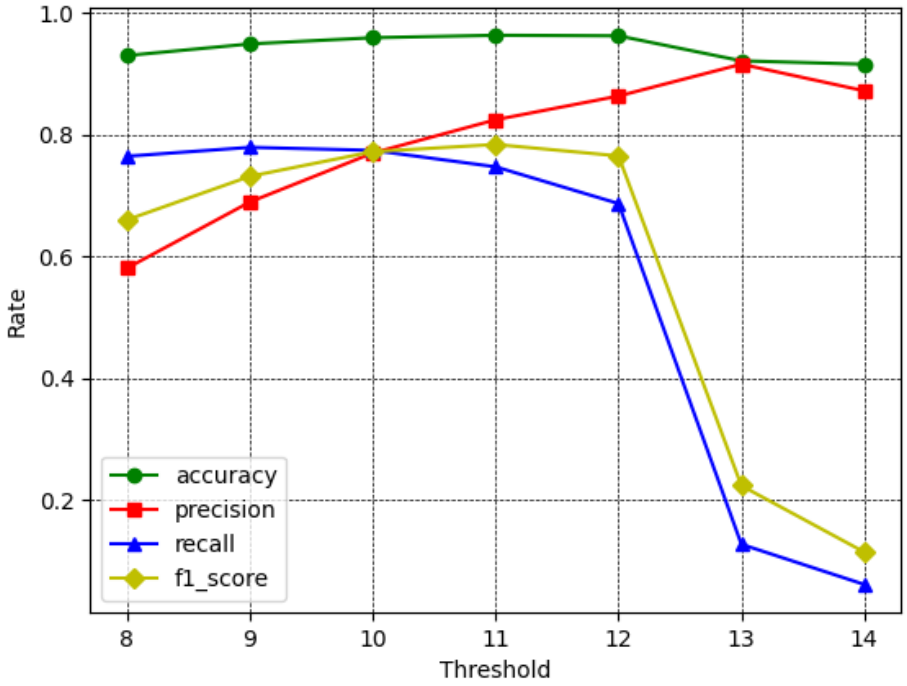
- Xây dựng hàm tính xác suất có điều kiện cho mỗi bigram để phục vụ cho việc phát hiện và sửa lỗi chính tả.

- Phát hiện lỗi chính tả: Áp dụng mô hình Bigram kết hợp phương pháp tra từ điển để phát hiện lỗi chính tả trong văn bản. Ở giai đoạn này cũng đồng thời tiến hành việc chọn "ngưỡng" cho mô hình dựa vào tập dữ liệu sai chính tả được xây dựng tại **mục 4.2.4**. Một "ngưỡng" phù hợp sẽ giúp mô hình đạt được hiệu suất tốt nhất có thể.

- Sửa lỗi chính tả: Khi phát hiện được lỗi chính tả, tiếp dụng sử dụng mô hình Bigram và phương pháp tra cứu từ điển kết hợp các thuật toán Khoảng cách soạn thảo và Chuỗi con chung dài nhất để có thể đưa ra các gợi ý sửa lỗi phù hợp.

- Đánh giá mô hình: Đánh giá hiệu suất của mô hình trong việc phát hiện và sửa lỗi chính tả bằng cách so sánh các kết quả của mô hình trên tập dữ liệu đã được đánh dấu lỗi chính tả theo các độ đo Precision, Recall và F1-score.

4.3.1 Xác định "ngưỡng" cho mô hình Bigram



Hình 11: Độ chính xác của mô hình khi "ngưỡng" thay đổi.

Khóa luận thực hiện việc xác định "ngưỡng" cho mô hình Bigram dựa trên tần suất xuất hiện của các bigram trong tập ngữ liệu (*corpus*). Nếu một bigram nhỏ hơn một "ngưỡng" chỉ định thì coi bigram đó có tồn tại lỗi sai chính tả. Trong quá trình thực nghiệm phát hiện lỗi chính tả, "ngưỡng" có điểm Precision, Recall và F1-score hợp lý thì sẽ được chọn.

Việc chọn "ngưỡng" phụ thuộc vào dữ liệu huấn luyện, vì vậy với dữ liệu mà khóa luận sử dụng thì "ngưỡng" được chọn là C = 10.

4.3.2 Kết quả thực nghiệm giai đoạn phát hiện lỗi

Test	Số câu	Số token	Số token lỗi	Số token lỗi được phát hiện	Số token lỗi được phát hiện đúng	Precision	Recall	F1-Score	Accuracy
1	36644	933983	82896	83365	64195	77%	77.44%	77.22%	95.94%
2	14100	154780	12006	13479	9852	73.09%	83.06%	77.32%	96.26%
3	7220	78184	7269	8745	5546	63.42%	76.30%	69.26%	93.70%
4	3519	39589	2782	3047	2224	72.99%	79.94%	76.30%	96.51%

Hình 12: Kết quả thực nghiệm phát hiện lỗi.

Khóa luận đánh giá hiệu suất của mô hình phát hiện lỗi chính tả dựa vào các giá trị TP, FP, TN và FN:

- TP (*True Positive*): Số lỗi chính tả mà mô hình phát hiện chính xác.
- FP (*False Positive*): Trường hợp mà mô hình phát hiện lỗi xảy ra sai lầm. Mô hình gán nhãn "sai" cho những token thực tế không phải là lỗi.
- TN (*True Negative*): Trường hợp mô hình đúng đắn. Mô hình gán đúng nhãn cho các token đúng.
- FN (*False Negative*): Trường hợp mô hình phát hiện xảy ra sai lầm. Mô hình bỏ sót các token thực tế là lỗi chính tả.

Các giá trị trên sẽ được sử dụng để tính toán các chỉ số đánh giá mô hình Precision, Recall, F1-score, Accuracy:

- Precision = $\frac{TP}{TP+FP}$

- Recall = $\frac{TP}{TP+FN}$
- F1-score = $\frac{2*Precision*Recall}{Precision+Recall}$
- Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$

4.3.3 Kết quả thực nghiệm giai đoạn sửa lỗi

Test	Số câu	Số token	Số token lỗi	Số lỗi phát hiện	Số lỗi được sửa chính xác	Số lỗi được sửa không chính xác	Precision
1	36644	933983	82896	83365	53699	29666	64.41%
2	14100	154780	12006	13479	8488	4991	62.97%
3	7220	78184	7269	8745	5344	3401	61.11%
4	3519	39589	2782	3047	1857	1190	60.94%

Hình 13: Kết quả thực nghiệm tự động sửa lỗi.

Test	Số câu	Số token	Số token lỗi	Số lỗi phát hiện	Số lỗi được sửa chính xác	Số lỗi được sửa không chính xác	Precision
1	36644	933983	82896	83365	60828	22537	72.96%
2	14100	154780	12006	13479	9480	3999	70.32%
3	7220	78184	7269	8745	5995	2750	68.55%
4	3519	39589	2782	3047	2126	921	69.77%

Hình 14: Kết quả thực nghiệm sửa lỗi theo từ gợi ý.

Khóa luận sử dụng chỉ số Precision để đánh giá độ hiệu quả của mô hình sửa lỗi chính tả. Giá trị Precision càng cao thì mô hình sửa lỗi càng chính xác và hạn chế các lỗi sửa sai.

- Precision = Số lỗi được sửa chính xác / Số lỗi mô hình phát hiện

Kết quả thực nghiệm tại Hình 13 là kết quả trong trường hợp phương án sửa lỗi là từ có thứ hạng cao nhất trong danh sách các từ đề xuất.

Tuy nhiên, nếu đánh giá độ hiệu quả của mô hình sửa lỗi trong trường hợp một lỗi chính tả được cho là sửa chính xác nếu phương án sửa nó nằm trong 5 vị trí đầu tiên của danh sách các từ đề xuất thì khóa luận thu được kết quả thực nghiệm như Hình 14.

Một số ví dụ về trường hợp sai chính tả mô hình phát hiện và tự động sửa lỗi:

- Câu đầu vào: Chính phủ yêu cầu điều **chỉn** lương , phụ cấp giáo viên.
- Kết quả: Chính phủ yêu cầu điều **chỉnh** lương , phụ cấp giáo viên.

- Câu đầu vào: Bộ **sương** như cánh tay quái vật dạt vào bờ biển.
- Kết quả: Bộ **xương** như cánh tay quái vật dạt vào bờ biển.

- Câu đầu vào: Bốn đề **suất** thúc đẩy tăng trưởng tại 21 nền kinh tế APEC.
- Kết quả: Bốn đề **xuất** thúc đẩy tăng trưởng tại 21 nền kinh tế APEC.

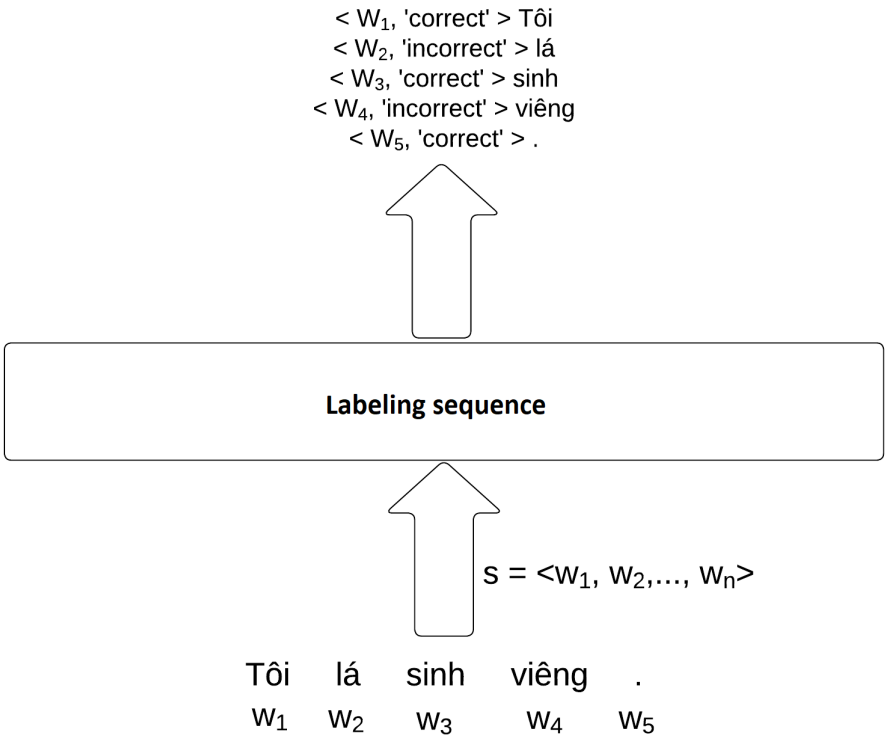
- Câu đầu vào: Thổ Nhĩ Kỳ được **xen** là đồng minh **khến** Mỹ và NATO đau đầu.
- Kết quả: Thổ Nhĩ Kỳ được **xem** là đồng minh **khiến** Mỹ và NATO đau đầu.

- Câu đầu vào: Quyết định của FIFA cấm rượu bia tại nơi **tiễn** ra các trận dấu World Cup 2022 khiến **mot** số du khách chỉ trích , nhưng **dc** nhiều CĐV ở Qatar ủng hộ.

- Kết quả: Quyết định của FIFA cấm rượu bia tại nơi **diễn** ra các trận dấu World Cup 2022 khiến **một** số du khách chỉ trích , nhưng **do** nhiều CĐV ở Qatar ủng hộ.

- Câu đầu vào: Động **co** làm bằng gốm mềm dẻo có thể vận hành ở nhiệt độ cao hơn nhiều so với động **co7** hợp kim **tuyền** thống và tiết kiệm nhiên **lieeju** hơn.
- Kết quả: Động **cơ** làm bằng gốm mềm dẻo có thể vận hành ở nhiệt độ cao hơn nhiều so với động **cơ** hợp kim tuyền thống và tiết kiệm nhiên **liệu** hơn.
- Câu đầu vào: Vài tuần sau đó, ông bị ói và **lôn** mửa.
- Kết quả: Vài tuần sau đó, ông bị **đi và nôn** mửa.

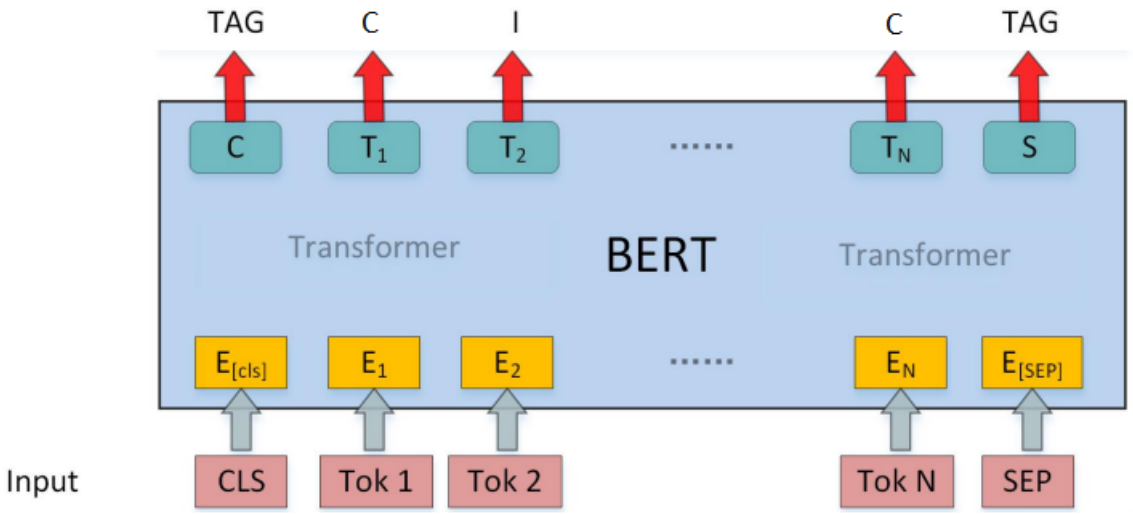
Phát hiện lỗi chính tả dựa trên mô hình học sâu



Hình 15: Minh họa gán nhãn chuỗi ứng dụng trong phát hiện lỗi chính tả.

Gán nhãn chuỗi (*sequence labeling*) là một bài toán trong lĩnh vực xử lý ngôn ngữ tự nhiên, trong đó mỗi thành phần (từ, cụm từ,...) trong một chuỗi dữ liệu được gán một nhãn tương ứng. Nhiệm vụ của gán nhãn chuỗi là dự đoán và gán nhãn cho mỗi thành phần trong chuỗi dựa trên ngữ cảnh của chúng. Có một số hướng tiếp cận cho bài toán gán nhãn chuỗi như mô hình CRF (*Conditional Random Fields*) hay mô hình Bộ nhớ dài-ngắn hạn (*Long-Short Term Memory*),... trong đó hướng tiếp cận hiện đại sử dụng mô hình BERT cho kết quả vượt trội hơn. Gán nhãn chuỗi sử dụng mô hình BERT để học các mẫu trong dữ liệu huấn luyện và dự đoán nhãn cho chuỗi đầu vào mới.

Bài toán phát hiện lỗi chính tả có thể được xem như một bài toán gán nhãn chuỗi , trong đó mỗi từ trong câu được gán nhãn là 'correct' nếu là từ đúng chính tả và được gán là 'incorrect' nếu nó là từ sai chính tả.



Hình 16: Bài toán gán nhãn chuỗi với mô hình BERT.

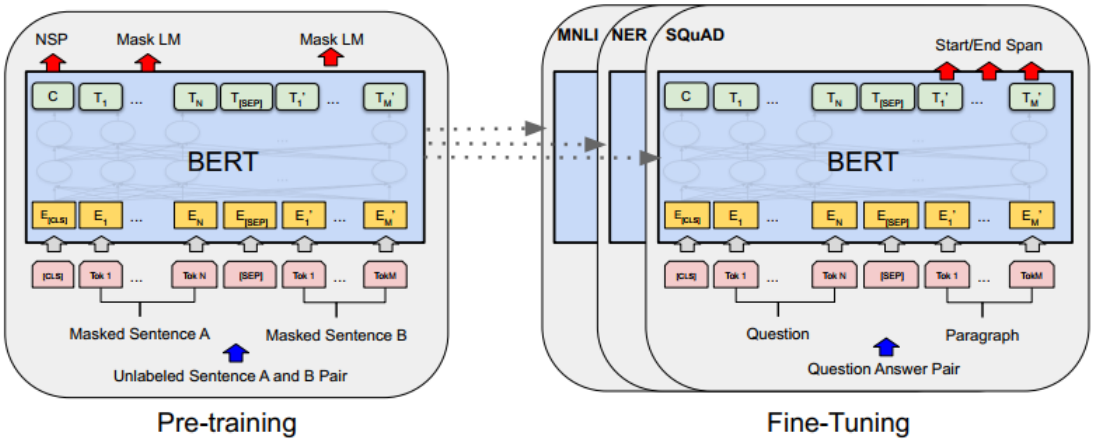
5.1 Mô hình BERT

BERT (*Bidirectional Encoder Representation from Transformer*), mô hình biểu diễn từ theo hai chiều ứng dụng kỹ thuật Transformer. BERT được thiết kế để huấn luyện trước (*pre-train*) các biểu diễn hai chiều từ văn bản chưa được gán nhãn bằng cách xem xét ngữ cảnh cả từ trái sang phải và từ phải sang trái, do đó nó được gọi là "biểu diễn hai chiều" (*bidirectional representation*) [8]. Đó chính là điểm khác biệt của BERT so với các mô hình trước đây khi chúng chỉ xem xét ngữ cảnh một chiều.

Theo bài báo [8], BERT đơn giản về mặt khái niệm và mạnh mẽ về mặt thực nghiệm. BERT đạt được kết quả tốt nhất trên 11 tác vụ xử lý ngôn ngữ tự nhiên, bao gồm đẩy độ chính xác trên tập GLUE lên 80,5% (cải thiện tuyệt đối 7,7%), độ chính xác trên tập MultiNLI lên 86,7% (cải thiện tuyệt đối 4,6%), với tập dữ liệu SQuAD v.1.1 F1-Score đạt 93.2% (cải thiện 1.5%) và SQuAD v2.0 F1-Score đạt 83.1% (cải thiện 5.1%)

Quá trình huấn luyện trước (*pre-training*) BERT là giai đoạn mà mô hình được huấn luyện trước trên một lượng lớn dữ liệu văn bản không giám sát. Mục tiêu của giai đoạn này là giúp mô hình hiểu và biểu diễn ngữ cảnh của các từ trong văn bản. Pre-train BERT bao gồm hai tác vụ chính: Masked Language Model (MLM) và Next Sentence Prediction (NSP) [8].

- **MLM:** Một số từ (15% các token) trong câu được chọn và bị che đi (masked) một cách ngẫu nhiên. Mô hình BERT dự đoán lại các từ được che dựa trên các từ không bị che và ngữ cảnh xung quanh.
- **NSP:** BERT đưa ra hai câu liên tiếp và dự đoán xem chúng có phải là câu tiếp theo của nhau hay không. Cụ thể, cặp câu đầu vào của mô hình là A và B sao cho 50% của B là câu theo sau A, được gán nhãn là "IsNext" và 50% đó nếu là một câu ngẫu nhiên từ ngữ liệu, tức không liên quan đến A, được gán nhãn là "NotNext".



Hình 17: Quá trình pre-train và fine-tuning của BERT [8].

Tại Hình 17, [CLS] là một kí tự đặc biệt được thêm vào ở đầu câu. [CLS] được sử dụng để biểu diễn toàn bộ câu đầu vào. [SEP] là một token đặc biệt được sử dụng để phân tách giữa các câu trong đầu vào. [SEP] được sử dụng để chỉ ra sự kết thúc của một câu, ví dụ phân tách "questions/answers" [8].

Sau khi hoàn thành quá trình huấn luyện trước (*pre-training*), mô hình BERT có thể được tinh chỉnh (*fine-tune*) đối với mỗi tác vụ cụ thể. Ở giai đoạn này, mô hình được huấn luyện trên dữ liệu có nhãn (*labeled data*) của tác vụ muốn áp dụng BERT. Quá trình tinh chỉnh điều chỉnh các tham số của mô hình để tối ưu hóa cho mục tiêu cụ thể của tác vụ đó như gán nhãn chuỗi, phân loại văn bản, nhận diện thực thể, dịch máy,... Quá trình này được thực hiện bằng cách huấn luyện mô hình trên tập dữ liệu được gán nhãn của tác vụ cụ thể và điều chỉnh tham số dựa trên mức độ sai số (*loss*) của mô hình.

Một mô hình tiếp nối của BERT là PhoBERT. Trong khi BERT được huấn luyện trên dữ liệu tiếng Anh thì PhoBERT được huấn luyện trên dữ liệu tiếng Việt. Điều này đồng nghĩa với việc BERT được tối ưu hóa cho việc xử lý văn bản tiếng Anh và PhoBERT được tối ưu hóa cho văn bản tiếng Việt.

Mặc dù BERT và PhoBERT có kiến trúc tương tự nhau để xử lý ngôn ngữ nhưng chúng có một số khác biệt nhỏ. PhoBERT được thiết kế để phù hợp với ngữ cảnh ngôn ngữ tiếng Việt và có một số điều chỉnh nhằm cải thiện khả năng biểu diễn ngữ nghĩa tiếng Việt trong một số tác vụ cụ thể.

Việc sử dụng BERT hay PhoBERT phụ thuộc vào ngôn ngữ của tác vụ. Nếu với tiếng Anh, BERT là ưu tiên hàng đầu. Ngược lại, với dữ liệu tiếng Việt, PhoBERT sẽ hiểu và xử lý ngôn ngữ tiếng Việt một cách hiệu quả hơn.

5.2 Thực nghiệm

Quá trình thực nghiệm:

- Cài đặt thư viện NERDA: Đây là một thư viện mã nguồn mở được phát triển cho tác vụ nhận dạng thực thể được đặt tên trong lĩnh vực xử lý ngôn ngữ tự nhiên. NERDA hỗ trợ tinh chỉnh các transformers cho các tác vụ NER đối với bất kỳ ngôn ngữ nào.

- Chuẩn bị dữ liệu: Chuẩn bị các tập dữ liệu đã được gán nhãn để

huấn luyện và kiểm thử mô hình.

- Xây dựng mô hình: Sử dụng thư viện NERDA để xây dựng mô hình cho bài toán phát hiện lỗi chính tả.
- Huấn luyện mô hình: Sử dụng tập dữ liệu huấn luyện đã chuẩn bị để huấn luyện mô hình.
- Đánh giá mô hình: Sử dụng tập dữ liệu kiểm thử để đánh giá hiệu suất của mô hình học sâu đối với bài toán phát hiện lỗi chính tả qua các độ đo Precision, Recall, F1-score.

5.2.1 Dữ liệu

	Số câu	Số token	Số token lỗi
Training set	29316	746956	66233
Validation set	3664	92326	8134
Test set	3664	94701	8529
Total	36644	933983	82896

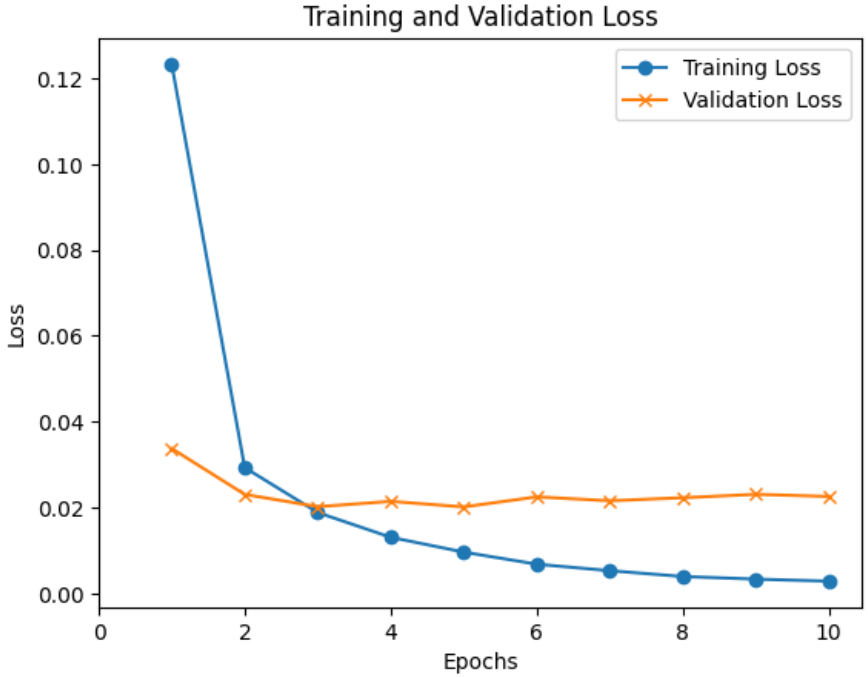
Hình 18: Dữ liệu huấn luyện mô hình học sâu.

Tập dữ liệu được xây dựng ở **mục 4.2.4** được chia thành ba tập dữ liệu như Hình 18: tập dữ liệu huấn luyện (*training dataset*), tập dữ liệu đánh giá (*validation dataset*) và tập dữ liệu kiểm thử (*test dataset*) lần lượt theo tỉ lệ 8:1:1. Tập dữ liệu huấn luyện bao gồm các mẫu dữ liệu có nhãn, tức các dữ liệu đã được gán nhãn đúng để mô hình học từ đó. Tập dữ liệu đánh giá được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Đây là một tập dữ liệu độc lập, không được sử dụng trong quá trình huấn luyện. Mô hình sẽ được đánh giá dựa trên tập đánh giá để đưa ra độ chính xác hay F1-score của mô hình. Tập dữ liệu kiểm thử được sử dụng để đánh giá mô hình sau khi đã hoàn thành quá trình huấn luyện.

5.2.2 Thực nghiệm

Phương pháp phát hiện lỗi chính tả với mô hình học sâu được thực nghiệm trên Google Colab sử dụng thư viện NERDA 1.0.0 với các thông số huấn luyện:

- Số epochs = 10
- Dropout = 0.1
- Learning rate = 0.00002
- Train batch size = 32
- Transformer = 'phobert-base'

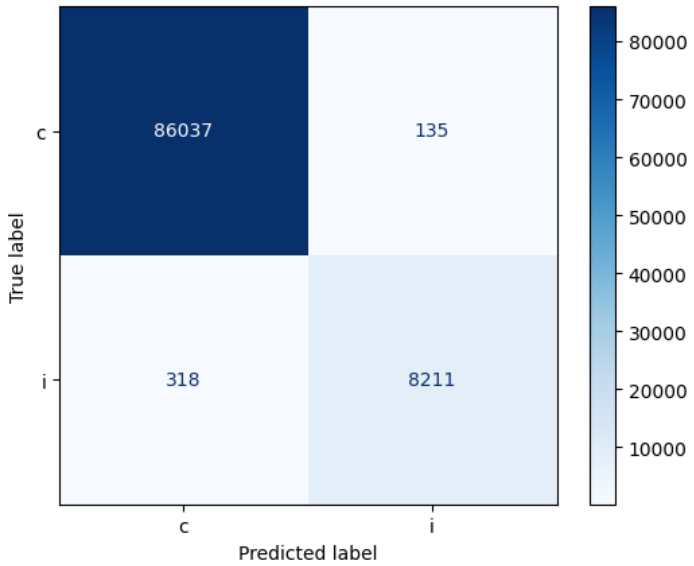


Hình 19: Sự thay đổi của loss sau mỗi epoch.

Sau khi huấn luyện mô hình, kết quả thu được tương đối tốt trên tập dữ liệu kiểm thử. Hình 20 thể hiện khả năng phát hiện lỗi chính tả của mô hình. Các độ đo hiệu suất của mô hình dựa trên confusion matrix:

- Precision: 98,38%

- Recall: 96,27%
- F1-score: 97,31%



Hình 20: Confusion matrix của mô hình học sâu trên tập dữ liệu kiểm thử.

Đối với tập dữ liệu kiểm thử, mô hình gần như phát hiện được chính xác các lỗi chính tả. Điều này đã cho thấy sự hiệu quả của mô hình học sâu trong việc phát hiện lỗi chính tả trong văn bản tiếng Việt. Tuy nhiên, để đạt được hiệu suất tốt nhất, mô hình cần được huấn luyện dựa trên một tập dữ liệu đa dạng và đủ lớn, và cần được cập nhật liên tục để đảm bảo khả năng áp dụng rộng rãi và đạt được độ chính xác cao.

Từ quá trình tìm hiểu, nghiên cứu, thực nghiệm mô hình học sâu và mô hình N-gram để giải quyết bài toán phát hiện lỗi chính tả, có thể nhận thấy:

- Mô hình học sâu có khả năng phát hiện lỗi tốt hơn so với mô hình N-gram.
- Mô hình học sâu có khả năng tổng hợp thông tin từ ngữ cảnh rộng hơn, bao gồm các từ xung quanh để dự đoán lỗi chính tả một cách chính xác hơn. Trong khi đó, mô hình N-gram xem xét các chuỗi

liên tiếp các từ, từ đó có thể phát hiện được lỗi chính tả dựa vào ngữ cảnh cục bộ của từ đó.

- Mô hình học sâu có cấu trúc phức tạp hơn mô hình N-gram, nó bao gồm nhiều lớp xử lý và nhiều tham số. Trong khi đó, mô hình N-gram có cấu trúc đơn giản hơn rất nhiều.
- Vì mô hình học sâu có cấu trúc phức tạp hơn nên sẽ tốn thời gian huấn luyện hơn so với mô hình N-gram.
- Mô hình học sâu và mô hình N-gram đều đòi hỏi tập dữ liệu huấn luyện đủ lớn để có thể đạt được hiệu suất tốt nhất.

	Mô hình Ngram	**Mô hình học sâu**
Số token	94701	94701
Số token lỗi	8529	8529
Số token lỗi được phát hiện	8578	8346
Số token lỗi được phát hiện chính xác	6548	8211
Precision	76.33%	98.38%
Recall	76.77%	96.27%
F1-Score	76.55%	97.31%

Hình 21: Hiệu suất phát hiện lỗi của mô hình N-gram so với mô hình học sâu.

	Mô hình Ngram	**Mô hình học sâu**
Số lỗi được phát hiện	8578	8346
Số lỗi được sửa chính xác	6293	6528
Số lỗi được sửa không chính xác	2285	1818
Precision	73.36%	78.22%

Hình 22: Hiệu suất sửa lỗi được phát hiện từ mô hình N-gram so với mô hình học sâu.

Tóm lại, mô hình N-gram và mô hình học sâu đều có ưu điểm và hạn chế riêng trong bài toán phát hiện lỗi chính tả. Nhưng mô hình học sâu thường vượt trội hơn mô hình N-gram bởi độ hiệu quả và tính linh hoạt của nó.

Kết luận

Kết quả đạt được:

Tìm hiểu và ứng dụng được mô hình ngôn ngữ N-gram vào thực tế, cụ thể là xây dựng chương trình "Phát hiện và sửa lỗi văn bản tiếng Việt".

Khóa luận đã xây dựng một tập dữ liệu gồm các văn bản tiếng Việt từ các nguồn thông tin điện tử chính thống và sử dụng nó để huấn luyện mô hình ngôn ngữ N-gram. Bên cạnh đó, khóa luận cũng xây dựng một tập dữ liệu sai chính tả dựa vào các quy tắc để có thể đánh giá độ hiệu quả của mô hình một cách chính xác nhất.

Sau khi khóa luận tiến hành thực nghiệm mô hình đối với tập dữ liệu huấn luyện và dữ liệu kiểm thử, các kết quả ban đầu mà mô hình N-gram đạt được khá khả quan. Giai đoạn phát hiện lỗi đạt kết quả 77.22% (độ đo F1-score). Giai đoạn sửa lỗi đạt kết quả 72.96% (độ đo Precision). Kết quả này cho thấy hệ thống phát hiện và sửa lỗi chính tả bằng mô hình ngôn ngữ N-gram có khả năng trong việc nhận diện và đề xuất sửa lỗi chính tả cho các từ sai chính tả trong văn bản. Hệ thống đã phát hiện được đa số các lỗi chính tả và đưa ra những gợi ý sửa lỗi phù hợp dựa trên xác suất xuất hiện của các từ và cấu trúc ngôn ngữ trong tập dữ liệu huấn luyện.

Khóa luận cũng tìm hiểu và thực nghiệm giai đoạn phát hiện lỗi chính tả theo hướng tiếp cận mô hình học sâu để so sánh độ hiệu quả so với mô hình N-gram. Kết quả thực nghiệm cho thấy mô hình học sâu có hiệu suất vượt trội so với mô hình N-gram. Giai đoạn phát hiện lỗi sử dụng mô hình học sâu đạt kết quả 97.31% (độ đo F1-score). Từ đó có thể mở rộng thêm nhiều hướng tiếp cận để giải quyết bài toán "Phát hiện và sửa lỗi văn bản tiếng Việt".

Hạn chế còn tồn tại:

Dựa trên tập dữ liệu kiểm thử, có thể thấy rằng tỉ lệ sửa lỗi chính xác còn chưa cao. Nếu từ sai chính tả không có trong tập dữ liệu huấn

luyện hoặc có nhiều cách viết khác nhau, hệ thống có thể gặp khó khăn trong việc đưa ra gợi ý sửa lỗi một cách chính xác nhất.

Dữ liệu huấn luyện: Mô hình ngôn ngữ N-gram được xây dựng trên tập dữ liệu huấn luyện tiếng Việt, vì vậy dữ liệu huấn luyện chưa đủ đa dạng và phong phú để đáp ứng tất cả các trường hợp và biến thể ngôn ngữ có thể xuất hiện trong văn bản tiếng Việt có thể dẫn đến việc mô hình không nhận diện hoặc không sửa chính xác các lỗi trong một số trường hợp.

Tốc độ chương trình chưa đủ nhanh, đặc biệt khi áp dụng cho các văn bản dài, điều này có thể hạn chế sự ứng dụng thực tế của mô hình, đặc biệt trong các hệ thống thời gian thực hoặc trên các nền tảng có tài nguyên tính toán hạn chế.

Việc phát hiện và sửa lỗi chính tả bị hạn chế trong các trường hợp lỗi liên quan đến cấu trúc ngôn ngữ như: lỗi sai cấu trúc câu, lỗi lặp từ,...

Hướng phát triển:

Tối ưu tốc độ chương trình.

Mở rộng dữ liệu huấn luyện nhằm cải thiện độ hiệu quả của mô hình ngôn ngữ N-gram.

Nghiên cứu và phát triển chương trình dựa vào các mô hình học sâu để phát hiện và sửa các lỗi chính tả, ngữ pháp, cấu trúc câu, lỗi sử dụng từ và các lỗi khác trong văn bản tiếng Việt.

Nghiên cứu cách tích hợp chương trình phát hiện và sửa lỗi văn bản vào các ứng dụng như trình soạn thảo văn bản, ứng dụng di động hoặc hệ thống chatbot. Điều này có thể hỗ trợ người dùng trong việc soạn thảo và giao tiếp bằng tiếng Việt.

Tài liệu

[1] Aminul Islam, Diana Inkpen (2009), "Real-Word Spelling Correction using Google Web 1T 3-grams", *Conference on Empirical Methods in Natural Language Processing*, Volume 3, pp. 1241–1249.

[2] A. V. Aho, D. S. Hirschberg and J. D. Ullman (1976), "Bounds on the complexity of the longest common subsequence problem", *Journal of the ACM*, Volume 23, Issue 1, pp. 1–12.

[3] Cong Duy Vu Hoang, Ai Ti AW (2012), "An Unsupervised and Data-Driven Approach for Spell Checking in Vietnamese OCR-scanned Texts", Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data (Hybrid2012), *EACL 2012*, pp. 36–44.

[4] Daniel Hládek, Ján Staš and Matúš Pleva (2020), "Survey of Automatic Spelling Correction", *Electronics 2020*, Volume 9, Issue 10, pp. 1670.

[5] Daniel Jurafsky, James H. Martin (2023), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Youcanprint.

[6] Gonzalo Navarro (2001), "A Guided Tour to Approximate String Matching", *ACM Computing Surveys*, Volume 33, Issue 1, pp. 31–88.

[7] Indu, Prerna (2016), "A Comparative Study of Different Longest Common Subsequence Algorithms", *Indu et al. International Journal of Recent Research Aspects ISSN: 2349-7688*, Vol. 3, Issue 2, pp. 65-69.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2019), "BERT: Pre-training of Deep Bidirectional

Transformers for Language Understanding", arXiv:1810.04805v2 [cs.CL].

[9] Nguyen Thi Xuan Huong, Tran-Thai Dang, The-Tung Nguyen and Anh-Cuong Le (2015), "Using Large N-gram for Vietnamese Spell Checking", *Advances in Intelligent Systems and Computing*, Volume 326, pp. 617-627.

[10] Phuong H. Nguyen, Thuan D. Ngo, Dung A. Phan, Thu P. T. Dinh and Thang Q. Huynh (2008), "Vietnamese spelling detection and correction using Bi-gram, Minimum Edit Distance, SoundEx algorithms with some additional heuristics", *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, Ho Chi Minh City, Vietnam, pp. 96-102, doi: 10.1109/RIVF.2008.4586339.

[11] Ritika Mishra, Navjot Kaur (2013), "A Survey of Spelling Error Detection and Correction Techniques", *International Journal of Computer Trends and Technology*, Volume 4, Issue3, pp. 372-374.

[12] `https://en.wikipedia.org/wiki/Longest_common_subsequence`, truy cập lần cuối 24 tháng 02 năm 2023.

[13] Lưu Tuấn Anh, http://viet.jnlp.org/kien-thuc-co-ban-ve-xu-ly-ngon-ngu-tu-nhien-mo-hinh-ngon-ngu, truy cập lần cuối 29 tháng 01 năm 2012.

[14] VSpell-TDK Development, `https://vspell.com/home/Specifications`, truy cập lần cuối 13 tháng 10 năm 2015.