Natural Language Processing

- Nguyễn Trung Hiếu –

Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh

Khoa CNTT

Email: hieuanthonydisward@gmail.com

Bài 3: Phân đoạn từ

Vấn đề: Trong một số ngôn ngữ(Tiếng Trung, tiếng Nhật, tiếng Thái và tiếng Việt), không sử dụng khoảng trắng để phân tách các từ, có nghĩa là một chuỗi kí tự có thể được hiểu là các tổ hợp từ khác nhau.

Phân đoạn từ là quá trình xác định các ranh giới từ này. Mục đích của nó, như đã nêu trong văn bản, là *loại bỏ sự mơ hồ trong ranh giới từ*. Bằng cách phân đoạn chính xác văn bản thành các từ riêng lẻ, các hệ thống NLP có thể thực hiện các nhiệm vụ khác như:

- **Gắn thẻ từ loại (Part-of-speech tagging):** Xác định vai trò ngữ pháp của mỗi từ (danh từ, động từ, tính từ, v.v.).
- **Phân tích cú pháp (Syntactic analysis):** Hiểu cấu trúc câu.
- **Phân tích ngữ nghĩa (Semantic analysis):** Hiểu nghĩa của văn bản.
- **Dịch máy (Machine translation):** Dịch văn bản sang ngôn ngữ khác

1. Các trường hợp của sự mơ hồ trong ranh giới từ

Bún chả ngon

- Bún/ chả ngon : Rice noodle/ isn’t delicious
- Bún chả/ ngon : Rice noodle with grilled meat rolls/ is delicious

Cột điện cao thế

- Cột điện cao_thế: Power pole high_power
- Cột điện / cao thế: Power pole is too high

Hổ mang bò lên núi

- Hổ_mang / bò lên núi: Cobra crawls up the mountain
- Hổ/ mang bò lên núi: Tiger brings cow to the mountain

Theo từ điển tiếng Việt Vietlex: có hơn 40.000 từ, trong đó:

- 15.69% từ trong từ điển là đơn âm tiết
- 70.72% từ ghép có 2 âm tiết
- 13.59% từ ghép >= 3 âm tiết
- 1.04% từ ghép >= 4 âm tiết

3. Cấu trúc từ tiếng Việt

Từ đơn:

Từ đơn là từ chỉ có một âm tiết, những từ này không thể phân chia thành các đơn vị nhỏ hơn có nghĩa.

Từ ghép:

Từ ghép là từ gồm nhiều âm tiết (hai âm tiết trở lên). Đặc điểm chính là các âm tiết này có *quan hệ ngữ nghĩa* với nhau – chúng đóng góp vào ý nghĩa chung của từ.

Phân loại từ ghép:

Từ ghép đẳng lập (Coordinated Compound Word)

Các thành phần cấu tạo (âm tiết/hình vị) có mối quan hệ ngang hàng và độc lập về mặt ý nghĩa. Chúng thường là các từ đồng nghĩa hoặc các khái niệm liên quan.

Ví dụ*: chợ búa: chợ* (nơi buôn bán) + *búa* (dụng cụ/bạo lực). Từ ghép này chỉ một khu chợ ồn ào, hỗn loạn hoặc một nơi tranh cãi, xung đột. Cả *chợ* và *búa* đều đóng góp ngang nhau vào ý nghĩa chung về một nơi ồn ào, náo động.

Lưu ý: Các phần được kết hợp với nhau để tạo ra một ý nghĩa rộng hơn hoặc sắc thái hơn, nhưng không phần nào phụ thuộc vào phần nào.

Từ ghép chính phụ (Main-Support Compound Word)

Một thành phần là thành phần *chính*, mang ý nghĩa cốt lõi, trong khi thành phần còn lại là thành phần *phụ*, bổ nghĩa, chuyên biệt hóa hoặc phân loại thành phần chính. Thành phần phụ phụ thuộc vào thành phần chính về mặt ý nghĩa trong từ ghép.

Ví dụ:

- *tàu hoả: tàu* (phương tiện) + *hoả* (lửa). *Tàu* là thành phần chính (phương tiện), và *hoả* (lửa) xác định loại phương tiện (tàu chạy bằng lửa/hơi nước).
- *đường sắt: đường* (lối đi) + *sắt* (kim loại). *Đường* là thành phần chính (lối đi), và *sắt* (kim loại) xác định vật liệu của lối đi (đường ray).
- *xấu bụng: xấu* (không tốt) + *bụng* (bên trong/tâm địa). *Xấu* là thành phần chính (không tốt), và *bụng* chỉ bên trong/tâm địa, nghĩa là "xấu bụng" hoặc "nham hiểm".
- *tốt mã: tốt* (tốt) + *mã* (vẻ bề ngoài). *Tốt* là thành phần chính (tốt), và *mã* chỉ vẻ bề ngoài, nghĩa là "đẹp trai" hoặc "có ngoại hình tốt".

Lưu ý: Thành phần phụ làm rõ hoặc bổ nghĩa cho ý nghĩa của thành phần chính. Thành phần chính thường có thể đứng một mình như một từ, trong khi thành phần phụ thường không thể (hoặc có nghĩa khác khi đứng một mình).

Từ láy(Repeated word):

Từ láy là từ mà một thành phần ngữ âm (âm tiết hoặc một phần âm tiết) được lặp lại, nhưng có sự biến đổi. Một từ đơn được lặp lại cũng tạo thành từ láy. Từ láy tạo ra hiệu ứng âm thanh đặc biệt, thường mang tính biểu cảm, nhấn mạnh, hoặc miêu tả trạng thái.

Đặc điểm:

- **Lặp lại:** Có sự lặp lại về âm thanh.

- **Biến đổi:** Sự lặp lại này thường đi kèm với sự biến đổi về âm đầu, vần, hoặc thanh điệu.

Ví dụ: Lung linh, xinh xắn

Biến thể của từ:

Biến thể của từ là sự thay đổi tạm thời hoặc hình thức "nói" của từ.

a. Rút gọn từ dài thành từ ngắn

- **Ví dụ:** *ki-lô-gam → ki lô/kí lô*. Đây là cách nói tắt thông dụng trong giao tiếp hàng ngày.
- **Điểm quan trọng:** Cách rút gọn này giúp giao tiếp nhanh chóng và tiện lợi hơn.

b. Tạm thời phá vỡ cấu trúc của từ, phân phối lại các yếu tố cấu tạo từ với các yếu tố từ các từ khác:

- **Ví dụ:**
  - *Lo khổ sở → lo khổ lo sở.*
  - *Cười ngặt nghẽo → cười ngặt cười nghẽo*.
  - *Danh lợi + ham chuộng → ham danh chuộng lợi*.
- **Đặc điểm:**
  - Thường xuất hiện trong khẩu ngữ, văn nói, hoặc văn thơ dân gian.
  - Tạo hiệu ứng nhấn mạnh, hài hước, hoặc tăng tính biểu cảm.
  - Phá vỡ cấu trúc thông thường của từ hoặc cụm từ.
- **Điểm quan trọng:** Đây là một hiện tượng ngôn ngữ thú vị, thể hiện sự linh hoạt và sáng tạo trong cách sử dụng ngôn ngữ.

Cụm từ đa từ (Multi-word expressions):

- **Định nghĩa:** Đây là các cụm từ bao gồm nhiều từ đơn, nhưng được coi như một đơn vị từ vựng duy nhất (single-word expressions) vì chúng thường biểu thị một ý nghĩa thống nhất và được sử dụng như một từ.
- **Ví dụ:** "bởi vì". Mặc dù bao gồm hai từ "bởi" và "vì", nhưng cụm từ "bởi vì" mang nghĩa "because" (bởi vì) và được sử dụng như một liên từ. Các ví dụ khác có thể kể đến như "tuy nhiên", "mặc dù", "cho nên"...
- **Điểm quan trọng:** Việc coi các cụm từ đa từ như một đơn vị từ vựng giúp cho việc xử lý ngôn ngữ tự nhiên (NLP) và phân tích cú pháp trở nên hiệu quả hơn.

Tên riêng (Proper name):

- **Định nghĩa:** Tên của người và chức danh được coi là một đơn vị từ vựng (lexical unit).
- **Ví dụ:** "Nguyễn Văn A" (tên người), "Chủ tịch nước" (chức danh). Mặc dù bao gồm nhiều từ, nhưng "Nguyễn Văn A" được coi là một đơn vị để xác định một cá nhân cụ thể, tương tự với "Chủ tịch nước" để chỉ một chức vụ.
- **Điểm quan trọng:** Điều này giúp phân biệt tên riêng với các cụm từ thông thường và giúp cho việc nhận dạng thực thể (Named Entity Recognition) trong NLP được chính xác hơn.

Các mẫu hình thường gặp (Regular patterns):

- **Định nghĩa:** Các mẫu hình này bao gồm các cách diễn đạt liên quan đến số và thời gian, thường tuân theo một cấu trúc nhất định.
- **Ví dụ:**
    - **Số:** "hai mươi mốt", "ba trăm linh năm", "một nghìn chín trăm chín mươi chín". Các cách diễn đạt số này tuân theo quy tắc nhất định về cách ghép các đơn vị số.

    - **Thời gian:** "ba giờ chiều", "mười giờ kém mười lăm", "ngày mười lăm tháng tám". Các cách diễn đạt thời gian này cũng tuân theo quy tắc về thứ tự và cách sử dụng các đơn vị thời gian.
- **Điểm quan trọng:** Việc nhận biết các mẫu hình này giúp cho việc phân tích và hiểu nghĩa của các biểu thức liên quan đến số và thời gian được dễ dàng hơn

Các phương pháp phân tích từ vựng:

Phương pháp dựa trên từ điển (Dictionary-based approach):

- **Nguyên lý:** Phương pháp này sử dụng một từ điển (hoặc danh sách từ vựng) làm cơ sở để xác định các từ trong văn bản. Khi gặp một chuỗi ký tự, hệ thống sẽ tra cứu trong từ điển để xem chuỗi đó có tồn tại như một từ hay không.
- **Ưu điểm:** Đơn giản, dễ thực hiện, hiệu quả đối với các ngôn ngữ có từ vựng tương đối ổn định và ít biến đổi.
- **Nhược điểm:** Khó xử lý các từ mới (từ mượn, từ lóng, từ địa phương), các từ ghép, từ láy, và các biến thể của từ. Hiệu suất phụ thuộc rất nhiều vào chất lượng và độ bao phủ của từ điển. Ví dụ, nếu từ điển không chứa từ "selfie" thì phương pháp này sẽ không nhận diện được từ đó.
- **Ví dụ trong phân đoạn từ:** Để phân đoạn câu tiếng Việt "tôihọcbài", phương pháp này sẽ tra từ điển để tìm các từ có thể tạo thành câu này. Nếu từ điển chứa "tôi", "học", và "bài", hệ thống sẽ phân đoạn thành "tôi học bài".

Phương pháp dựa trên học máy (Machine learning-based approach):

- **Nguyên lý:** Phương pháp này sử dụng các thuật toán học máy để học từ dữ liệu huấn luyện (training data) về cấu trúc và đặc điểm

của từ trong ngôn ngữ. Sau khi được huấn luyện, mô hình có thể dự đoán và phân loại các từ trong văn bản mới.

- **Ưu điểm:** Khả năng xử lý tốt các từ mới, từ ghép, từ láy, và các biến thể của từ. Có khả năng tự động thích nghi với các thay đổi trong ngôn ngữ.
- **Nhược điểm:** Đòi hỏi lượng lớn dữ liệu huấn luyện chất lượng cao. Hiệu suất phụ thuộc vào thuật toán và chất lượng dữ liệu. Khó giải thích được quyết định của mô hình (black box).
- **Ví dụ trong phân đoạn từ:** Một mô hình học máy được huấn luyện trên một tập dữ liệu lớn các câu tiếng Việt đã được phân đoạn sẽ học được các quy tắc về cách kết hợp các âm tiết thành từ. Khi gặp câu "tôihọcbài", mô hình sẽ dự đoán và phân đoạn thành "tôi học bài" dựa trên những gì đã học. Các mô hình phổ biến bao gồm Hidden Markov Models (HMM), Conditional Random Fields (CRF), và các mạng nơ-ron (neural networks).

Kết hợp các phương pháp (A combination of these methods):

- **Nguyên lý:** Phương pháp này kết hợp ưu điểm của cả hai phương pháp trên. Ví dụ, sử dụng từ điển để xử lý các từ phổ biến và sử dụng học máy để xử lý các từ mới hoặc phức tạp.
- **Ưu điểm:** Cải thiện độ chính xác và khả năng xử lý của hệ thống. Tận dụng được nguồn lực từ điển sẵn có và khả năng học hỏi của học máy.
- **Ví dụ:** Trong một hệ thống phân đoạn từ, từ điển có thể được sử dụng để phân đoạn nhanh các từ đơn giản. Sau đó, mô hình học máy sẽ được sử dụng để xử lý các trường hợp phức tạp hơn như từ ghép, từ láy hoặc các từ không có trong từ điển.

4. Thuật toán Longest word matching(Ghép từ dài nhất)

Đây là thuật toán theo phương pháp dựa trên từ điển.

Yêu cầu:

Từ điển (Dictionary): Cần có một từ điển chứa danh sách các từ hợp lệ trong ngôn ngữ.

Chuỗi đầu vào đã được phân đoạn bởi dấu câu và khoảng trắng (The input string has been segmented by punctuations and spaces): Điều này có nghĩa là thuật toán giả định rằng các dấu câu và khoảng trắng đã được xử lý trước đó, và đầu vào chỉ còn là một chuỗi liên tục các âm tiết cần được phân đoạn thành từ.

Ý tưởng:

Sử dụng thuật toán tham lam(Greedy Algorithm)

Phân tích từ trái sang phải hoặc từ phải sang trái, lấy từ dài nhất có thể cho đến khi hết âm tiết: Đây là cốt lõi của thuật toán. Thuật toán sẽ duyệt chuỗi đầu vào theo một hướng (từ trái sang phải hoặc từ phải sang trái) và cố gắng ghép các âm tiết thành từ dài nhất có thể tìm thấy trong từ điển.

Độ phức tạp tính toán (Computational complexity): O(n . V)

- **n: số âm tiết trong chuỗi đầu vào (#syllables in the input string)**
- **V: số từ trong từ điển (#words in the dictionary)**
- Độ phức tạp này cho thấy thời gian thực hiện của thuật toán tỉ lệ tuyến tính với số âm tiết trong chuỗi đầu vào và số từ trong từ điển. Điều này có nghĩa là nếu chuỗi đầu vào dài hơn hoặc từ điển lớn hơn, thời gian thực hiện sẽ tăng lên tương ứng.

Ví dụ:

Giả sử câu là "họcsinhlớp1" và từ điển chứa: "học", "sinh", "học sinh", "lớp", "lớp 1", "1".

- **Phân tích từ trái sang phải:**
    1. Bắt đầu từ "học". Tra từ điển, thấy "học" là một từ hợp lệ.
    2. Tiếp theo là "sinh". Tra từ điển, thấy "sinh" là một từ hợp lệ.
    3. Tiếp theo là "lớp". Tra từ điển, thấy "lớp" là một từ hợp lệ.
    4. Cuối cùng là "1". Tra từ điển, thấy "1" là một từ hợp lệ.
    5. Kết quả phân đoạn (sai): "học sinh lớp 1".
- **Phân tích từ trái sang phải (với việc ưu tiên từ dài nhất):**
    1. Bắt đầu từ "học".
    2. Kiểm tra "học sinh". Tra từ điển, thấy "học sinh" là một từ hợp lệ.
    3. Tiếp theo là "lớp 1". Tra từ điển, thấy "lớp 1" là một từ hợp lệ.
    4. Kết quả phân đoạn (đúng): "học sinh lớp 1".

Thuật toán

Start

Khởi tạo:

1. Cho các đầu vào[w_0 w_1 .. w_n-1]
2. Words <- []
3. S <-0

Vòng lặp

4. E <- n
5. Khi [w_s .. w_e] không phải là từ: e <- e-1
6. Words <- words + [w_s … w_e]
7. S <- e+1
8. Nếu e<n: Quay lại bước 4

Kết thúc

9. Trả về chuỗi đã được phân đoạn thành từ

End.

Giải thích từng bước trong sơ đồ:

1. **Given an input [w_0 w_1 ... w_(n-1)] (Cho một đầu vào [w_0 w_1 ... w_(n-1)])**: Đây là bước khởi đầu. w_0, w_1, ..., w_(n-1) đại diện cho các âm tiết hoặc ký tự trong chuỗi đầu vào cần được phân đoạn. n là tổng số âm tiết/ký tự. Ví dụ, với câu "tôihọcbài", ta có w_0 = tôi, w_1 = học, w_2 = bài và n = 3.
2. **words ← [] (từ ← []):** Khởi tạo một danh sách rỗng words để lưu trữ các từ đã được phân đoạn.
3. **s ← 0 (s ← 0):** Khởi tạo biến s (start - bắt đầu) bằng 0. s là chỉ số bắt đầu của từ hiện tại trong chuỗi đầu vào.
4. **e ← n (e ← n):** Khởi tạo biến e (end - kết thúc) bằng n. e là chỉ số kết thúc của từ hiện tại trong chuỗi đầu vào. Ban đầu, e được đặt bằng độ dài của chuỗi, tức là chúng ta đang xem xét toàn bộ chuỗi làm một từ tiềm năng.
5. **When [w_s ... w_e] is not a word: e ← e-1 (Khi [w_s ... w_e] không phải là một từ: e ← e-1):** Đây là vòng lặp chính của thuật toán. Nó kiểm tra xem chuỗi con từ w_s đến w_e ([w_s ... w_e]) có tồn tại trong từ điển hay không. Nếu không, nó sẽ giảm e đi 1, tức là rút ngắn chuỗi con được xem xét từ bên phải. Vòng lặp này tiếp tục cho đến khi tìm thấy một từ hợp lệ trong từ điển hoặc khi e bằng s.
6. **words ← words + [w_s ... w_e] (từ ← từ + [w_s ... w_e]):** Nếu chuỗi con [w_s ... w_e] là một từ hợp lệ trong từ điển, nó sẽ được thêm vào danh sách words.

7. **s ← e + 1 (s ← e + 1):** Cập nhật chỉ số bắt đầu s bằng e + 1. Điều này có nghĩa là chúng ta sẽ bắt đầu tìm từ tiếp theo từ vị trí ngay sau từ vừa tìm được.
8. **If e < n: Go to step 4 (Nếu e < n: Quay lại bước 4):** Kiểm tra xem chúng ta đã xử lý hết chuỗi đầu vào hay chưa. Nếu e vẫn nhỏ hơn n, tức là vẫn còn phần chuỗi chưa được xử lý, thuật toán sẽ quay lại bước 4 để tiếp tục tìm từ tiếp theo.
9. **Return the string that has been segmented into words (Trả về chuỗi đã được phân đoạn thành các từ):** Khi e bằng n, tức là chúng ta đã xử lý toàn bộ chuỗi đầu vào, thuật toán sẽ trả về danh sách words chứa các từ đã được phân đoạn.

Ví dụ với "tôihọcbài":

1. Đầu vào: "tôihọcbài"
2. words = [], s = 0, e = 3
3. Kiểm tra "tôihọcbài" trong từ điển. Giả sử từ điển không có từ này.
4. e = 2. Kiểm tra "tôihọc" trong từ điển. Giả sử không có.
5. e = 1. Kiểm tra "tôi" trong từ điển. Giả sử có.
6. words = ["tôi"]
7. s = 2, e = 3
8. Kiểm tra "họcbài" trong từ điển. Giả sử không có.
9. e = 2. Kiểm tra "học" trong từ điển. Giả sử có.
10. words = ["tôi", "học"]
11. s = 3, e = 3
12. Kiểm tra "bài" trong từ điển. Giả sử có.
13. words = ["tôi", "học", "bài"]

14. Kết quả: "tôi học bài"

Điểm cần lưu ý:

- Thuật toán này là *tham lam* vì nó luôn chọn từ dài nhất tại mỗi bước mà không xem xét các khả năng phân đoạn khác. Điều này có thể dẫn đến kết quả không tối ưu trong một số trường hợp.
- Hiệu suất của thuật toán phụ thuộc vào chất lượng và độ bao phủ của từ điển.
- Việc xử lý các từ ghép, từ láy và các hiện tượng ngôn ngữ phức tạp khác có thể gặp khó khăn với thuật toán này.

Ưu điểm (Advantages):

- **Dễ cài đặt (Simple to implement):** Thuật toán này tương đối đơn giản về mặt logic và dễ dàng để lập trình. Nó chỉ yêu cầu một từ điển và một vài phép so sánh chuỗi.
- **Độ phức tạp hợp lý (Reasonable complexity):** Như đã nói trước đó, độ phức tạp tính toán của thuật toán là O(n . V), với n là số âm tiết trong chuỗi đầu vào và V là số từ trong từ điển. Mặc dù phụ thuộc vào kích thước của từ điển, nhưng trong nhiều trường hợp, độ phức tạp này vẫn được coi là chấp nhận được.
- **Không yêu cầu dữ liệu huấn luyện (No training data required):** Đây là một ưu điểm lớn so với các phương pháp dựa trên học máy. Thuật toán "Ghép từ dài nhất" không cần một tập dữ liệu lớn đã được gán nhãn để huấn luyện. Nó chỉ cần một từ điển.

Nhược điểm (Disadvantages):

- **Phụ thuộc vào từ điển (Depend on the dictionary):** Hiệu suất của thuật toán phụ thuộc hoàn toàn vào chất lượng và độ bao phủ của từ điển.
    - Nếu từ điển thiếu một từ nào đó, thuật toán sẽ không thể phân đoạn chính xác. Ví dụ, nếu từ điển không có từ "selfie" (một

từ mượn tiếng Anh), thuật toán sẽ không nhận diện được từ này trong một văn bản tiếng Việt.

    - Từ điển càng lớn thì thời gian tra cứu càng lâu, ảnh hưởng đến hiệu suất.

- **Vấn đề mơ hồ chưa được giải quyết (The ambiguity issue has not been resolved):** Đây là nhược điểm chính của thuật toán. Do tính chất tham lam của nó, thuật toán luôn chọn từ dài nhất tại mỗi bước, mà không xem xét các khả năng phân đoạn khác. Điều này có thể dẫn đến kết quả phân đoạn sai trong các trường hợp có nhiều cách phân đoạn hợp lệ.
    - **Ví dụ:** Xét câu "cáihộpđỏ" với từ điển chứa các từ: "cái", "hộp", "đỏ", "cái hộp".
        - Nếu thuật toán chỉ xét từng từ đơn, kết quả sẽ là "cái hộp đỏ".
        - Nếu thuật toán ưu tiên từ dài nhất, kết quả sẽ là "cái hộp đỏ" (vì "cái hộp" là một từ trong từ điển).
        - Trong trường hợp này, kết quả là đúng, nhưng nếu từ điển chỉ chứa "hộp" và không chứa "cái hộp", kết quả sẽ là "cái hộp đỏ" (vẫn đúng).

Bài tập 1:

Thực hiện thuật toán ghép từ dài nhất bằng python:

Các mẫu thử:

– Thời khóa biểu đang được cập nhật

The timetable is being updated

– Môn học xử lý ngôn ngữ tự nhiên

Natural language processing course

– Con ngựa đá con ngựa đá

The horse kicks the stone horse

– Học sinh học sinh học

Student learn biology

```
def valid_word(word, dictionary):
    return word in dictionary

def segment_longest_words(input_string, dictionary):
    words = []
    s = 0
    n = len(input_string)

    while s < n:
        e = n
        while e > s and not valid_word(input_string[s:e], dictionary):
            e -= 1
        if e > s:
            words.append(input_string[s:e])
        else:
            words.append(input_string[s])
            s += 1
    return " ".join(words)
```

Phương pháp phân đoạn từ đơn giản nhất(Simplest word segmentation)

Phát hiện các mẫu chung như tên riêng, viết tắt, số, ngày tháng, địa chỉ email, URL, v.v. bằng cách sử dụng biểu thức chính quy.

Phương pháp: Sử dụng các biểu thức chính quy (regular expressions) để định nghĩa các mẫu tìm kiếm tương ứng với các loại thông tin này.

Tìm cách chia đoạn văn bản thành các từ có nghĩa bằng cách chọn chuỗi âm tiết dài nhất có thể tìm thấy trong từ điển.

Phương pháp:

- Đối với mỗi vị trí bắt đầu trong văn bản, tìm kiếm chuỗi âm tiết dài nhất từ vị trí đó mà vẫn có trong từ điển.

- Trong trường hợp có nhiều cách chia khác nhau, chọn cách chia có số lượng từ ít nhất.

Ưu điểm: Nhanh chóng và hiệu quả đối với các mẫu có cấu trúc rõ ràng.

Hạn chế: Không thể xử lý các trường hợp phức tạp hơn, đặc biệt là các từ ghép, từ nhiều nghĩa.

Hướng giải quyết: Liệt kê tất cả để chọn khả năng tốt nhất

Biểu thức chính quy trong phân đoạn tự.

Là phần để nối với chuỗi

Một số kí tự đặc biệt:

*: bất kì chuỗi kí tự nào, kể cả chuỗi rỗng

X: ít nhất một kí tự

+: Chuỗi trong dấu ngoặc xuất hiện ít nhất 1 lần

Ví dụ:
- Email: x@x(.x)+ : "tên@tên miền(.tên miền)"

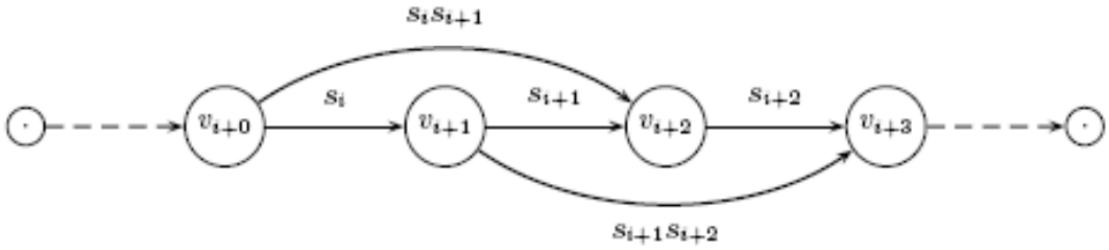
- dir *.txt

- '*John -> 'John', "Ajohn", "Decker John"

Biểu thức chính quy được sử dụng thường xuyên trong:

- Phân tách kí tự
- Xác thực dữ liệu
- Xử lý chuỗi
- Trích xuất thông tin

Chọn phương pháp tốt nhất



Biểu diễn chuỗi bằng dãy âm tiết:

- **S1, S2, ..., Sn:** Đây là các âm tiết trong chuỗi cần phân đoạn.
- **Ví dụ:** Chuỗi "xinchao" sẽ được biểu diễn thành dãy âm tiết "xin chao".

Trường hợp mơ hồ:

- **3 âm tiết liên tiếp:** Đây là trường hợp phức tạp nhất vì có nhiều cách phân đoạn khác nhau.
- **s1s2 và s2s3 là từ:** Cả hai cặp âm tiết liền kề đều có thể tạo thành từ, gây ra sự mơ hồ trong việc phân đoạn.

Đồ thị có hướng:

- **G = (V, E):** G là đồ thị có hướng, V là tập hợp các đỉnh, E là tập hợp các cạnh.
- **V = {V0, V1, ..., Vn, Vn+1}:** Mỗi đỉnh Vi tương ứng với vị trí giữa âm tiết Si và Si+1.
- **Cạnh:** Nếu các âm tiết từ Si+1 đến Sj tạo thành một từ, thì đồ thị sẽ có một cạnh nối từ đỉnh Vi đến đỉnh Vj.

```
1: V ← ∅;
2: for i = 0 to n + 1 do
3: V ← V □ {vi};
4: end for
5: for i = 0 to n do
```

```
6:      for j = i to n do
7:          If (accept(AW, si · · · sj)) then
8:              E ← E □ {(vi, vj+1)};
9:          end if
10:   end for
11: end for
12: return G = (V,E);
```

1.Khởi tạo:

- V ← ∅;: Khởi tạo một tập hợp rỗng V để lưu trữ các đỉnh của đồ thị.

2. Tạo các đỉnh:

- for i = 0 to n + 1 do: Lặp qua các giá trị từ 0 đến n+1.
- V ← V ∪ {vi};: Trong mỗi lần lặp, thêm một đỉnh mới vi vào tập hợp V. Điều này tạo ra một tập hợp các đỉnh đại diện cho các vị trí giữa các âm tiết.

3. Thêm các cạnh:

- for i = 0 to n do: Lặp qua các giá trị từ 0 đến n.
- for j = i to n do: Lặp qua các giá trị từ i đến n.
- if (accept(AW, si · · · sj)) then: Kiểm tra xem chuỗi con từ âm tiết si đến sj có phải là một từ hợp lệ hay không bằng cách sử dụng hàm accept(AW, si · · · sj).
- E ← E ∪ {(vi, vj+1)};: Nếu chuỗi con là một từ hợp lệ, thêm một cạnh có hướng từ đỉnh vi đến đỉnh vj+1 vào tập hợp các cạnh E. Cạnh này đại diện cho một phân đoạn từ có thể tại vị trí giữa sj và sj+1.
- end if
- end for
- end for

4. Trả về đồ thị:

- return G = (V,E);: Trả về đồ thị đã xây dựng G dưới dạng một bộ gồm tập hợp các đỉnh V và tập hợp các cạnh E.

5. Giải quyết vấn đề mơ hồ(Ambiguity resolution)

Mỗi chuỗi từ có một xác suất: Mỗi cách phân đoạn một câu thành các từ sẽ có một xác suất nhất định.

Xác suất được tính dựa trên ngữ cảnh: Xác suất của một từ phụ thuộc vào các từ trước đó trong câu.

Mô hình n-gram: Để đơn giản hóa việc tính toán, người ta thường sử dụng mô hình n-gram, trong đó n là số lượng từ trước đó được xem xét. Ví dụ, bigram (n=2) xem xét từ trước đó, trigram (n=3) xem xét hai từ trước đó

Xác suất của chuỗi s:

$$\mathrm{P(s)} = \prod_{i=1}^{m} P(w_i|w_i^{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1})$$

$\prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1})$: Đây là xác suất có điều kiện của từ thứ i (wi) khi biết n-1 từ trước đó (wi-n+1, wi-n+2, ..., wi-1). Nói cách khác, xác suất xuất hiện của một từ phụ thuộc vào ngữ cảnh của n-1 từ đứng trước nó.

$\prod$ **(tích từ i=1 đến m):** Ký hiệu này có nghĩa là ta sẽ nhân lần lượt các xác suất từ từ đầu tiên đến từ cuối cùng của chuỗi.

n: Là một tham số quyết định số lượng từ trước đó được xem xét khi tính xác suất. Ví dụ:

- **n=2:** Mô hình bigram, chỉ xét từ trước đó.
- **n=3:** Mô hình trigram, xét hai từ trước đó.

Khi n = 2, tính toán xác suất tối đa(Maximum Likelihood-ML) cho bigram

$$P_{ML}(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} = \frac{c(w_{i-1}w_i)/N}{c(w_{i-1})/N}$$

P(wi|wi-1): Xác suất có điều kiện của từ wi khi biết từ trước đó là wi-1. Nói cách khác, đây là xác suất xuất hiện của từ wi sau khi đã biết từ wi-1.

PML(wi|wi-1): Xác suất tối đa của wi cho biết wi-1. Đây là ước tính xác suất dựa trên số lần xuất hiện của cặp từ (wi-1, wi) trong tập dữ liệu huấn luyện.

c(wi-1wi): Số lần xuất hiện của cặp từ (wi-1, wi) trong tập dữ liệu huấn luyện.

c(wi-1): Số lần xuất hiện của từ wi-1 trong tập dữ liệu huấn luyện.

N: Tổng số từ trong tập dữ liệu huấn luyện.

Nhưng có vấn đề:

Vấn đề khi tần số thấp: Khi số lần xuất hiện của một cặp từ rất nhỏ so với tổng số từ trong tập dữ liệu (c(s) << N), xác suất tính được có thể gần bằng 0. Điều này gây ra vấn đề khi sử dụng mô hình, vì các cặp từ hiếm gặp sẽ có xác suất rất thấp, ảnh hưởng đến kết quả dự đoán.

Giải quyết:

Để giải quyết vấn đề trên, người ta thường sử dụng các phương pháp làm mịn (smoothing) như:

- **Laplace smoothing:** Thêm một hằng số nhỏ vào cả tử số và mẫu số của phân số để tránh trường hợp xác suất bằng 0.
- **Kneser-Ney smoothing:** Sử dụng thông tin về tần suất xuất hiện của các từ đơn lẻ và các bigram khác để ước tính xác suất.

$$P(w_i|w_{i-1}) = \lambda_1 P_{ML}(w_i|w_{i-1}) + \lambda_2 P_{ML}(w_i)$$

$$P_{ML}(w_i) = \frac{c(w_i)}{N}$$

Với T = $\{s_1, s_2, \dots s_n\}$

$$P(T) = \sum_{i=1}^{n} P(s_i)$$

T: Một chuỗi từ gồm n từ.

P(T): Xác suất của chuỗi từ T.

P(si): Xác suất của từ thứ i trong chuỗi T.

Tính toán $\lambda_1\lambda_2$ (Để tối đa hóa)

$$L(\lambda_1\lambda_2) = \sum_{w_{i-1},w_i} C(w_{i-1}, w_i) \log_2 P(w_i|w_{i-1})$$

Với $\lambda_1+\lambda_2 = 1$ và $\lambda_1, \lambda_2 \geq 0$

Thuật toán: Tìm giá trị lamda

Algorithm 2. Finding Lamda value

$\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0.5;$

$\epsilon \leftarrow 0.01;$

repeat

$\widehat{\lambda}_1 \leftarrow \lambda_1, \quad \widehat{\lambda}_2 \leftarrow \lambda_2;$

$c_1 \leftarrow \sum_{w_{i-1}, w_i} \frac{C(w_{i-1}, w_i)\lambda_1 P_{ML}(w_i|w_{i-1})}{\lambda_1 P_{ML}(w_i|w_{i-1}) + \lambda_2 P_{ML}(w_i)};$

$c_2 \leftarrow \sum_{w_{i-1}, w_i} \frac{C(w_{i-1}, w_i)\lambda_2 P_{ML}(w_i)}{\lambda_1 P_{ML}(w_i|w_{i-1}) + \lambda_2 P_{ML}(w_i)};$

$\lambda_1 \leftarrow \frac{c_1}{c_1 + c_2}, \quad \lambda_2 \leftarrow 1 - \lambda_1;$

$\widehat{\epsilon} \leftarrow \sqrt{(\widehat{\lambda}_1 - \lambda_1)^2 + (\widehat{\lambda}_2 - \lambda_2)^2};$

until $(\widehat{\epsilon} \leq \epsilon);$

return $\lambda_1, \lambda_2;$

Khởi tạo:

- Gán giá trị ban đầu cho λ1 và λ2 là 0.5. Đây là một giá trị khởi đầu hợp lý, cho phép cả hai thành phần (bigram và unigram) có ảnh hưởng tương đương.
- Thiết lập một ngưỡng sai số rất nhỏ ε (ví dụ: 0.01). Ngưỡng này sẽ được sử dụng để kiểm tra sự hội tụ của thuật toán.

Lặp đi lặp lại cho đến khi hội tụ:

- **Tính toán các giá trị mới cho λ1 và λ2:**
  - Các giá trị mới của λ1 và λ2 được tính dựa trên các giá trị cũ và các số liệu thống kê từ tập dữ liệu (c(wi-1, wi), PML(wi|wi-1), PML(wi)).
  - Ý tưởng chính ở đây là điều chỉnh các giá trị λ1 và λ2 để tối ưu hóa việc ước tính xác suất, sao cho phù hợp với dữ liệu thực tế.

    - Các công thức tính c1 và c2 khá phức tạp, nhưng cơ bản là chúng đại diện cho các trọng số được sử dụng để cập nhật λ1 và λ2.

- **Kiểm tra điều kiện dừng:**
    - Tính toán độ chênh lệch giữa các giá trị mới và cũ của λ1 và λ2.
    - Nếu độ chênh lệch này nhỏ hơn ngưỡng ε, tức là các giá trị đã hội tụ, thuật toán dừng lại.

Trả về kết quả:

- Khi thuật toán hội tụ, các giá trị cuối cùng của λ1 và λ2 được trả về. Đây là các giá trị tối ưu mà thuật toán tìm được.

6. Phương pháp kết hợp tiếng Việt

Kết hợp nhiều kĩ thuật: Automat(mô hình hóa quy tắc ngữ pháp) + Regrex(biểu thức chính quy) + longest matching(Chọn ký tự dài nhất) + Probabilistic(Đánh giá khả năng một chuỗi kí tự là 1 từ)

Một vài công cụ để phân tách từ:

JvnSegmenter (Nguyễn Cẩm Tú) : CRF

http://jvnsegmenter.sourceforge.net

• VnTokenizer (Lê Hồng Phương)

https://github.com/phuonglh/vn.vitk

• Dongdu (Lưu Anh Tuấn): SVM

http://viet.jnlp.org/dongdu

• Pyvi (Trần Việt Trung) : https://github.com/trungtv/pyvi

• Word dictionaries:

• http://tratu.coviet.vn/tu-dien-lac-viet.aspx

- http://tratu.soha.vn/
- https://www.informatik.uni-leipzig.de/~duc/Dict/

Tham khảo:

Tham khảo từ slide môn xử lý ngôn ngữ tự nhiên của HUST

Nguồn Slide: https://github.com/hoanglong1712/Dai-Hoc-Bach-Khoa-Ha-Noi/tree/main/Natural%20Language%20Processing