ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# Tách từ tiếng Việt

Viện Công nghệ Thông tin và Truyền thông

# Tách từ

- Mục đích: xác định ranh giới của các từ trong câu.
- Là bước xử lý quan trọng đối với các hệ thống XLNNTN, đặc biệt là đối với các ngôn ngữ đơn lập, ví dụ: âm tiết Trung Quốc, âm tiết Nhật, âm tiết Thái, và tiếng Việt.
- Với các ngôn ngữ đơn lập, một từ có thể có một hoặc nhiều âm tiết.

➢ Vấn đề của bài toán tách từ là khử được sự nhập nhằng trong ranh giới từ.

# Từ vựng

- Tiếng Việt là ngôn ngữ không biến hình
- Từ điển từ tiếng Việt (Vietlex): >40.000 từ, trong đó:
  - 81.55% âm tiết là từ : từ đơn
  - 15.69% các từ trong từ điển là từ đơn
  - 70.72% từ ghép có 2 âm tiết
  - 13.59% từ ghép ≥ 3 âm tiết
  - 1.04% từ ghép ≥ 4 âm tiết

# Từ vựng

- Tiếng Việt là ngôn ngữ không biến hình
- Từ điển từ tiếng Việt (Vietlex): >40.000 từ

| Độ dài | # từ | % |
|---|---|---|
| 1 | 6,303 | 15.69 |
| 2 | 28,416 | 70.72 |
| 3 | 2,259 | 5.62 |
| 4 | 2,784 | 6.93 |
| 5 | 419 | 1.04 |
| Tổng | 40,181 | 100 |

Bảng 1. Độ dài của từ tính theo âm tiết

# Qui tắc cấu tạo từ tiếng Việt

- Từ đơn: *dùng một âm tiết* làm một từ.
  - Ví dụ: *tôi, bác, người, cây, hoa, đi, chạy, vì, đã, à, nhỉ, nhé...*
- Từ ghép: *tổ hợp* (ghép) các âm tiết lại, giữa các âm tiết đó có quan hệ về nghĩa với nhau.
  - Từ ghép đẳng lập. các thành tố cấu tạo có quan hệ bình đẳng với nhau về nghĩa.
    - Ví dụ: *chợ búa, bếp núc*
  - Từ ghép chính phụ. các thành tố cấu tạo này phụ thuộc vào thành tố cấu tạo kia. Thành tố phụ có vai trò phân loại, chuyên biệt hoá và sắc thái hoá cho thành tố chính.
    - Ví dụ: *tàu hoả, đường sắt, xấu bụng, tốt mã, ngay đơ, thẳng tắp, sưng vù...*

# Qui tắc cấu tạo từ tiếng Việt

- Từ láy: các yếu tố cấu tạo có thành phần ngữ âm được lặp lại; nhưng vừa lặp vừa biến đổi. Một từ được lặp lại cũng cho ta từ láy.
- Biến thể của từ: được coi là *dạng lâm thời biến động* hoặc *dạng "lời nói"* của từ.
  - Rút gọn một từ dài thành từ ngắn hơn
    - ki-lô-gam → ki lô/ kí lô
  - Lâm thời phá vỡ cấu trúc của từ, phân bố lại yếu tố tạo từ với những yếu tố khác ngoài từ chen vào. Ví dụ:
    - khổ sở → lo khổ lo sở
    - ngặt nghẽo → cười ngặt cười nghẽo
    - danh lợi + ham chuộng → ham danh chuộng lợi

# Qui tắc cấu tạo từ tiếng Việt

- Các diễn tả gồm nhiều từ (vd, “bởi vì”) cũng được coi là 1 từ
- Tên riêng: tên người và vị trí được coi là 1 đơn vị từ vựng
  - Các mẫu thường xuyên: số, thời gian

# Các hướng tiếp cận

- Tiếp cận dựa trên từ điển
- Tiếp cận dựa trên học máy
- Kết hợp hai phương pháp trên.

# Tách từ dựa trên từ điển

- Thuật toán so khớp từ dài nhất
- Yêu cầu:
  - Từ điển
  - Chuỗi đầu vào đã tách các dấu câu và âm tiết
- Tư tưởng: thuật toán tham lam
  - Đi từ trái sang phải hoặc từ phải sang trái, lấy các từ dài nhất có thể, dừng lại khi duyệt hết
  - Độ phức tạp tính toán: *O(n . V)*
    - *n: Số âm tiết trong chuỗi*
    - *V: Số từ trong từ điển*

# Tách từ dựa trên từ điển

- Thuật toán so khớp từ dài nhất

---

- **BẮT ĐẦU** — *khởi tạo*
- (1) Cho chuỗi đầu vào [*w_0 w_1 … w_n-1*]
- (2) *words* ← *[]*
- (3) *s* ← *0*

---

- (4) *e* ← *n* — *lặp*
- (5) Khi [*w_s … w_e*] **chưa là một từ**: *e* ← *e – 1*
- (6) *words* ← *words* + *[w_s … w_e]*
- (7) *s* ← *e* + *1*
- (8) Nếu *e* < *n*: Quay lại bước (4)

---

- (9) Lấy ra chuỗi đã tách từ *words* — *kết thúc*
- **KẾT THÚC**

---

# Thuật toán so khớp từ dài nhất

- Ưu điểm:
  - Cài đặt đơn giản
  - Độ phức tạp tính toán hợp lý
  - Không yêu cầu dữ liệu huấn luyện
- Nhược điểm:
  - Phụ thuộc vào từ điển
  - Chưa giải quyết được vấn đề nhập nhằng

# Bài tập

- Cài đặt thuật toán so khớp từ dài nhất trên Python
- Một số mẫu thử:
  - *Thời khóa biểu đang được cập nhật*
  - *Môn học xử lý ngôn ngữ tự nhiên*
  - *Ông già đi nhanh quá*
  - *Con ngựa đá con ngựa đá*
  - *Học sinh học sinh học*

```python
def tokenizer(text, dict, is_show=False):
    print ("input:", text)
    print ()
    input = text.split(" ") #[w_0, w_1,..., w_n-1]
    words = []
    s = 0
    while True:
        e = len(input)
        while e > s:
            tmp_word = input[s:e] # [w_s ... w_e]
            is_word = ""
            for item in tmp_word:
                is_word += item + " "
            is_word = is_word[:-1] #Loại bỏ dấu cách thừa ở cuối
            e -= 1
            # print (is_word)
            if is_word.lower() in dict:
                words.append(is_word) # words <- words + [w_s ... w_e]
                break
```

```
19                  break
20              if e == s:
21                  words.append(is_word) # words <- words + first_word
22                  break
23          if e >= len(input):
24              break
25          #Hiển thị quá trình tách từ
26          if is_show:
27              print("s =", s)
28              print("e =", e)
29              print(words[len(words) - 1])
30              print("-" * 100)
31          s = e + 1
32      output = ""
33      for item in words:
34          output += item.replace(" ", "_")
35          output += " "
36      output = output[:-1]
37      return output
```

```python
if __name__ == "__main__":
    ex1 = "thời khóa biểu đang được cập nhật"
    ex2 = "môn học xử lý ngôn ngữ tự nhiên"
    ex3 = "con ngựa đá con ngựa đá"
    ex4 = "học sinh học sinh học"

    #Từ điển
    dict = {"thời khóa biểu": 0,"đang": 1,"được": 2,"cập nhật": 3,
            "môn học": 4,"môn": 5; "học": 6,"xử lý": 7,"ngôn ngữ": 8
            "tự nhiên": 9,"con": 10, "con ngựa": 11, "ngựa": 12,
            "đá": 13, "học": 13, "học sinh": 14, "sinh học": 15,
            "dân tộc": 16, "viện trưởng": 17, "giáo viên": 18,
            "đạo diễn": 19,"xứ sở": 20, "nguồn lực": 21, "thủ đô": 2
            "số lượng": 23, "thuần nhất": 24, "môi giới": 25,
            "đơn giản": 26, "tiến bộ": 27, "chính sách": 28,
            "thường xuyên": 29, "tình yêu": 30; }

    test1 = tokenizer(ex2, dict)

    print ("output:", test1)
```

Có thể chạy online trên https://aivietnam.ai/

# Cách tách từ đơn giản

- Phát hiện các mẫu thông thường như tên riêng, chữ viết tắt, số, ngày tháng, địa chỉ email, URL,… sử dụng biểu thức chính qui
- Chọn chuỗi âm tiết dài nhất từ vị trí hiện tại và có trong từ điển, chọn cách tách có ít từ nhất

➢ Hạn chế: có thể đưa ra cách phân tích không đúng.

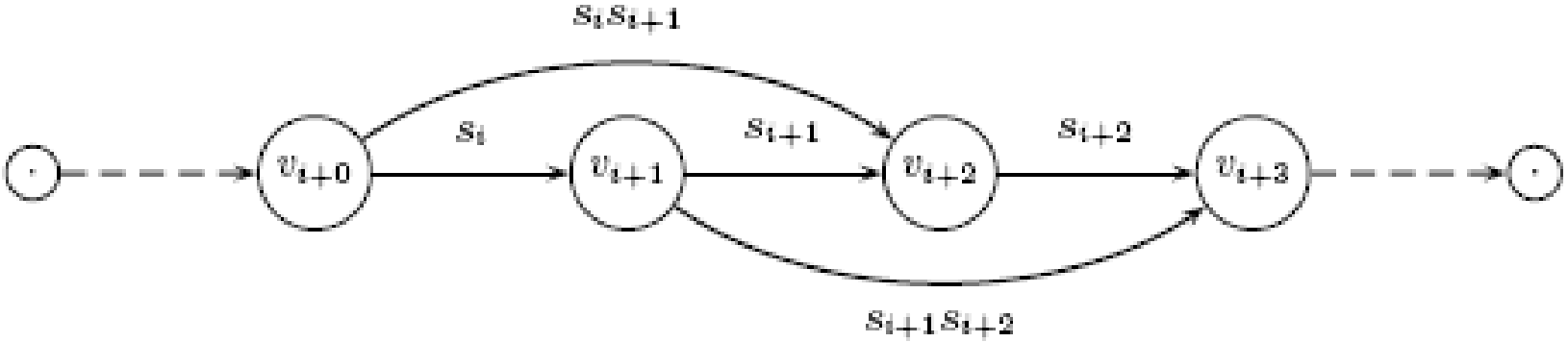
➢ Giải quyết: liệt kê tất,  có 1 chiến lược để chọn cách tách tốt nhất.

# Tách từ sử dụng biểu thức chính qui

- là một khuôn mẫu được so sánh với một chuỗi
- Các ký tự đặc biệt:
  - * - bất cứ chuỗi ký tự nào, kể cả không có gì
  - x – ít nhất 1 ký tự
  - + - chuỗi trong ngoặc xuất hiện ít nhất 1 lần
- Ví dụ:
  - Email: x@x(.x)+
  - dir *.txt
  - '*John' -> 'John', 'Ajohn', "Decker John"
- Biểu thức chính quy được sử dụng đặc biệt nhiều trong:
  * Phân tích cú pháp
  * Xác nhận tính hợp lệ của dữ liệu
  * Xử lý chuỗi
  * Trích rút thông tin

# Lựa chọn cách tách từ

- Biểu diễn đoạn bằng chuỗi các âm tiết $s_1\, s_2 \ldots s_n$
- Trường hợp nhập nhằng thường xuyên nhất là 3 từ liền nhau $s_1s_2s_3$ trong đó $s_1s_2$ và $s_2s_3$ đều là từ.




- BIểu diễn 1 đoạn bằng đồ thị có hướng tuyến tính $G = (V,E)$, $V = \{v_0, v_1, \ldots, v_n, v_{n+1}\}$
- Nếu các âm tiết $s_{i+1}, s_{i+2}, \ldots, s_j$ tạo thành 1 từ -> trong G có cạnh $(v_i,v_j)$
- Các cách tách từ = các đường đi ngắn nhất từ $v_0$ đến $v_{n+1}$

# Thuật toán

**Thuật toán 1. Xây dựng đồ thị cho chuỗi $s_1 s_2 \ldots s_n$**

*1:* $V \leftarrow \emptyset$;
2: **for** $i = 0$ **to** $n + 1$ **do**
3: $V \leftarrow V \cup \{v_i\}$;
4: **end for**
5: **for** $i = 0$ **to** $n$ **do**
6: **for** $j = i$ **to** $n$ **do**
7: **if** (accept($A_W$, $s_i \cdot \cdot \cdot s_j$)) **then**
8: $E \leftarrow E \cup \{(v_i, v_{j+1})\}$;
9: **end if**
10: **end for**
11: **end for**
12: **return** $G = (V,E)$;

accept($A$, $s$): automat $A$ nhận xâu vào $s$

# Phân giải nhập nhằng

- Xác suất xâu s:

$$P(s) = \prod_{i=1}^{m} P(w_i | w_1^{i-1}) \approx \prod_{i=1}^{m} P(w_i | w_{i-n+1}^{i-1})$$

- $P(w_i|w_1^{i-1})$: xác suất $w_i$ khi có i-1 từ trước đó
- n = 2: bigram; n = 3: trigram

# Phân giải nhập nhằng

- Khi n = 2, tính giá trị $P(w_i|w_{i-1})$ lớn nhất maximum likelihood (ML)

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

- c(s): số lần xâu s xuất hiện; N: tổng số từ trong tập luyện
- Khi dữ liệu luyện nhỏ hơn kích cỡ toàn bộ tập dữ liệu → P ~ 0
  - ➢ Sử dụng kỹ thuật làm trơn

# Kỹ thuật làm trơn

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

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

$P_{ML}(w_i) = c(w_i)/N$

- Với tập thử nghiệm T = $\{s_1, s_2, \ldots, s_n\}$, xác suất P(T) của tập thử:

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

# Xác định giá trị $\lambda_1$, $\lambda_2$

- Từ tập dữ liệu mẫu, định nghĩa $C(w_{i-1},w_i)$ là số lần $(w_{i-1}, w_i)$ xuất hiện trong tập mẫu. Ta cần chọn $\lambda_1$ $\lambda_2$ để làm cực đại giá trị

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

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

# Thuật toán

**Thuật toán 2.** Xác định giá trị $\lambda$

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

$\epsilon \leftarrow 0.01;$

**repeat**

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

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

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

$\lambda_1 \leftarrow \frac{c_1}{c_1 + c_2}, \quad \lambda_2 \leftarrow 1 - \widehat{\lambda}_1;$

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

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

**return** $\lambda_1, \lambda_2;$

# Cách tiếp cận lai

*<Phuong Le-Hong et al., A hybrid approach to word segmentation of Vietnamese texts, Proceedings of the 2nd International Conference on Language and Automat Theory and Applications, LATA 2008, Tarragona, Spain, 2008.>*

- Kết hợp phân tích automat hữu hạn + biểu thức chính quy + so khớp từ dài nhất + thống kê (để giải quyết nhập nhằng)

# Kết quả

- Sử dụng tập dữ liệu gồm 1264 bài trong báo Tuổi trẻ, có 507,358 từ
- Lấy $\varepsilon = 0.03$, các giá trị $\lambda$ hội tụ sau 4 vòng lặp

| Step | $\lambda_1$ | $\lambda_2$ | $\epsilon$ |
|---|---|---|---|
| 0 | 0.500 | 0.500 | 1.000 |
| 1 | 0.853 | 0.147 | 0.499 |
| 2 | 0.952 | 0.048 | 0.139 |
| 3 | 0.981 | 0.019 | 0.041 |
| 4 | 0.991 | 0.009 | 0.015 |

- Độ chính xác = số từ hệ thống xác định đúng/tổng số từ hệ thống xác định = 95%

# Một số công cụ 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
- Từ điển từ:
  - http://tratu.coviet.vn/tu-dien-lac-viet.aspx
  - http://tratu.soha.vn/
  - https://www.informatik.uni-leipzig.de/~duc/Dict/

**Bài tập**: cài đặt và chạy chương trình tách từ Pyvi

```python
from pyvi import ViTokenizer

ex1 = "thời khóa biểu đang được cập nhật"
ex2 = "môn học xử lý ngôn ngữ tự nhiên"
ex3 = "con ngựa đá con ngựa đá"
ex4 = "học sinh học sinh học"
ex5 = "Tách từ là bài toán nhận diện từ trong văn bản tiếng Việt"

if __name__ == "__main__":
    print (ViTokenizer.tokenize(ex5))
```