Word Segmentation

Lê Thanh Hương
School of Information and Communication Technology - HUST
Email: huonglt@soict.hust.edu.vn

Word Segmentation

- Purpose: determine word boundaries in a sentence.
- It is an important step for NLP systems, especially for monosyllable languages like Chinese, Japanese, Thai, and Vietnamese.
- In monosyllable languages, a word can have one or more syllables.

➢ The task of word segmentation is to eliminate ambiguity in word boundaries.

Some cases of ambiguity

- 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.

Some cases of ambiguity

- Quyết định liều tiêm cho trẻ 5-11 tuổi.

- Quyết_định liều_tiêm cho trẻ 5-11 tuổi.
- Deciding the dose for children 5-11 years old.

- Quyết_định liều / tiêm cho trẻ 5-11 tuổi.
- Risky decision / injection for children 5-11 years old

Vocabulary

- Vietnamese is an inflectional language
- Vietnamese word dictionary (Vietlex): >40,000 words, in which:
  - 81.55% syllables: monosyllable words
  - 15.69% words in the dictionary are monosyllable
  - 70.72% compound words with 2 syllables
  - 13.59% compound words ≥ 3 syllables
  - 1.04% compound words ≥ 4 syllables

Vocabulary

- Vietnamese word dictionary(Vietlex): >40.000 words

#syllables in a word	# words	%
1	6,303	15.69
2	28,416	70.72
3	2,259	5.62
4	2,784	6.93
5	419	1.04
Total	40,181	100

Table 1. Word length based on syllables

Structure of Vietnamese words

- Single word: a word consists of one syllable
  - E.g.: *tôi, bác, người, cây, hoa, đi, chạy, vì, đã, à, nhỉ, nhé...*
- Compound word : a word consists of several syllables. These syllables are semantically related.
  - Coordinated compound word (từ ghép đẳng lập): The structural elements have an equal relationship with each other in terms of meaning
    - E.g.: *chợ búa, bếp núc*
  - Main-support compound word (từ ghép chính phụ): One element depends on another. The supporting element has the role of classifying, specializing the main element.
    - E.g.: *tàu hoả, đường sắt, xấu bụng, tốt mã, ngay đơ, thẳng tắp, sưng vù...*

Structure of Vietnamese words

- Repeated word (từ láy): a phonetic component of the word is repeated; but iterates and transforms. A single word that is repeated also gives us a repeated word.
- Variation of a word : is a temporary modification or "speech" form of the word.
  - Shorten a long word to a shorter word
    - ki-lô-gam → ki lô/ kí lô
  - Temporarily break the structure of words, redistribute word-forming elements with elements from other words
    - Lo khổ sở → lo khổ lo sở
    - Cười ngặt nghẽo → cười ngặt cười nghẽo
    - danh lợi + ham chuộng → ham danh chuộng lợi

Structure of Vietnamese words

- Multi-word expressions (e.g., “bởi vì”) are also considered single-word expressions
- Proper name: name of person and position are considered as a lexical unit
- Regular patterns: number, time

Approaches

- Dictionary-based approach
- Machine learning-based approach
- A combination of these methods

Dictionary-based approach

- Longest word matching algorithm
- Requirement:
  - Dictionary
  - The input string has been segmented by punctuations and spaces
- Idea: greedy algorithm
  - Parse from left-to-right or right-to-left, taking the longest word possible until no syllable left
  - Computational complexity: O*(n . V)*
    - *n:* #syllables in the input string
    - *V: #words in the dictionary*

Dictionary-based approach

- Longest matching algorithm

Start

1. Given an input [w_0 w_1 … w_n-1] — Initiate
2. words ← []
3. s ← 0

4. e ← n
5. When [w_s … w_e] is not a word: e ← e-1
6. words ← words + [w_s … w_e] — Loop
7. s ← e+1
8. If e <n: Go to step 4

9. Return the string that has been segmented into words — End

End

Longest matching algorithm

- Advantages:
  - Simple to implement
  - Reasonable complexity
  - No training data required
- Disadvantages:
  - Depend on the dictionary
  - The ambiguity issue has not been resolved

Exercise 1

- Implement the longest word matching algorithm in Python
- Some test samples:
  - *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

```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] # Remove redundant spaces at the end
            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          # Display the word segmentation process
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
```

```
39  if __name__ == "__main__":
40      ex1 = "thời khóa biểu đang được cập nhật"
41      ex2 = "môn học xử lý ngôn ngữ tự nhiên"
42      ex3 = "con ngựa đá con ngựa đá"
43      ex4 = "học sinh học sinh học"
44
45      #Từ điển
46      dict = {"thời khóa biểu": 0,"đang": 1,"được": 2,"cập nhật": 3,
47              "môn học": 4,"môn": 5; "học": 6,"xử lý": 7,"ngôn ngữ": 8,
48              "tự nhiên": 9,"con": 10, "con ngựa": 11, "ngựa": 12,
49              "đá": 13, "học": 13, "học sinh": 14, "sinh học": 15,
50              "dân tộc": 16, "viện trưởng": 17, "giáo viên": 18,
51              "đạo diễn": 19,"xứ sở": 20, "nguồn lực": 21, "thủ đô": 22,
52              "số lượng": 23, "thuần nhất": 24, "môi giới": 25,
53              "đơn giản": 26, "tiến bộ": 27, "chính sách": 28,
54              "thường xuyên": 29, "tình yêu": 30; }
55
56      test1 = tokenizer(ex2, dict)
57
58      print ("output:", test1)
```

Exercise 3

- Add more words in the dictionary so that the code can analyze the following sentences. All possible words of those sentences should be added in the dictionary.
  - *Ông già đi nhanh quá*
  - *Bún chả ngon*
  - *Cột điện cao thế*
  - *Hổ mang bò lên núi*

Simplest word segmentation

- Detect common patterns like proper names, abbreviations, numbers, dates, email addresses, URLs, etc. using regular expressions
- Choose the longest sequence of syllables from the current position and in the dictionary, choose the solution with the fewest words

➢ Limitations: may return incorrect analysis.

➢ Solution: list all and have a strategy to choose the best solution

Regrex in word segmentation

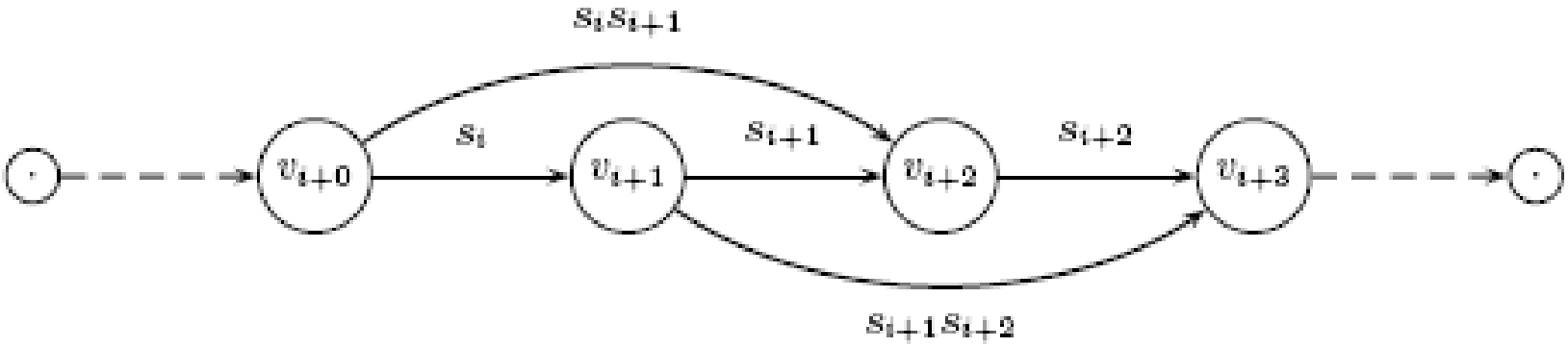
- is a pattern to map with a string
- Special characters:
- * : any string of characters, including nothing
- x : at least 1 character
- + : The string in brackets appears at least once
- E.g.:
  - Email: x@x(.x)+
  - dir *.txt
  - ‘*John’ -> ‘John’, ‘Ajohn’, “Decker John”
- Regrex is most often used in:
  - Syntactic parsing
  - Data validation
  - String processing
  - Information extraction

Exercise 3

- Extend the code to deal with proper names, numbers and datetime.
- Test with the following sentences:
  - *Tôi gặp An lúc 2pm.*
  - *Học sinh học môn sinh học vào tiết 3.*

Choosing the best solution

- Representing the input string by a sequence of syllables $s_1\ s_2 \ldots\ s_n$
- The most ambiguity case is 3 continuous syllables $s_1s_2s_3$ In which s_1s_2 and s_2s_3 are words



- Represent a string with a linear directed graph $G = (V,E)$, $V = \{v_0,\ v_1, \ldots, v_n, v_{n+1}\}$
- If syllables $s_{i+1}, s_{i+2}, \ldots, s_j$ form a word → G has an edge (v_i,v_j)
- Word segmentation solutions = the shortest paths from v_0 to v_{n+1}

Algorithm

Algorithm 1. Constructing a graph for the string $s_1 s_2 \ldots s_n$

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 accepts the input string s

Ambiguity resolution

- Probability of the string s:

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

- $P(w_i|w_1^{i-1})$: probability of w_i when the previous i-1 words are determined
- n = 2: bigram; n = 3: trigram

Ambiguity resolution

- When n = 2, compute the maximum likelihood (ML) of $P(w_i|w_{i-1})$

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

- c(s): #occurrences of s; N: #words in the training set
- When c(s) << N → P ~ 0
  - ➢ Using smoothing method

Smoothing method

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

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

- With $T = \{s_1, s_2, \ldots, s_n\}$:

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

Compute λ_1, λ_2

- Choose λ_1 λ_2 to maximize:

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

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

Algorithm

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 $\lambda_1, \lambda_2;$

Hybrid approach

<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.>

- Combine automat + regrex + longest matching + probabilistic (to solve ambiguity)

Results

- Using the dataset with 1264 articles in Tuổi trẻ journal, with 507,358 words
- With $\varepsilon = 0.03$, the values of λ converge after 4 loops

Step	λ_1	λ_2	ϵ
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

- Precision= #words correctly being predicted/ #words being predicted = 95%

Some segmentation tools

- *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/

Exercise: install and run 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))
```

Conditional Random Fields (CRFs): sequence annotation

- Ông già đi nhanh quá
- B_w B_w B_w B_w B_w
- Ông già đi nhanh quá
- B_w I_w B_w B_w B_w

- Features:
  - IsCapitalize
  - IsNumber
  - …..

SVM: classification

- Ông già đi nhanh quá

  0 1 1 1