Word Segmentation

Lê Thanh Hương
School of Information and Communication Technology
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 *hay* bún chả ngon
- Bún / chả ngon   Bún_chả / ngon
- Cột điện cao thế *hay* cột điện cao thế!
- Cột điện cao_thế   Cột điện / cao thế
- Mất mạng *hay* mất mạng
- Mất mạng   Mất mạng
- Hổ mang bò lên núi *hay* hổ mang bò lên núi
- Hổ_mang / bò lên núi   Hổ / mang bò lên núi
- Năm con hổ đang đến *hay* năm con hổ đang đến.
- Năm con hổ / đang đến   Năm con hổ / đang đến
- Quyết_định liều tiêm cho trẻ 5-11 tuổi.
- Quyết_định liều tiêm cho trẻ 5-11 tuổi.

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

    khổ sở → lo khổ lo sở

    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

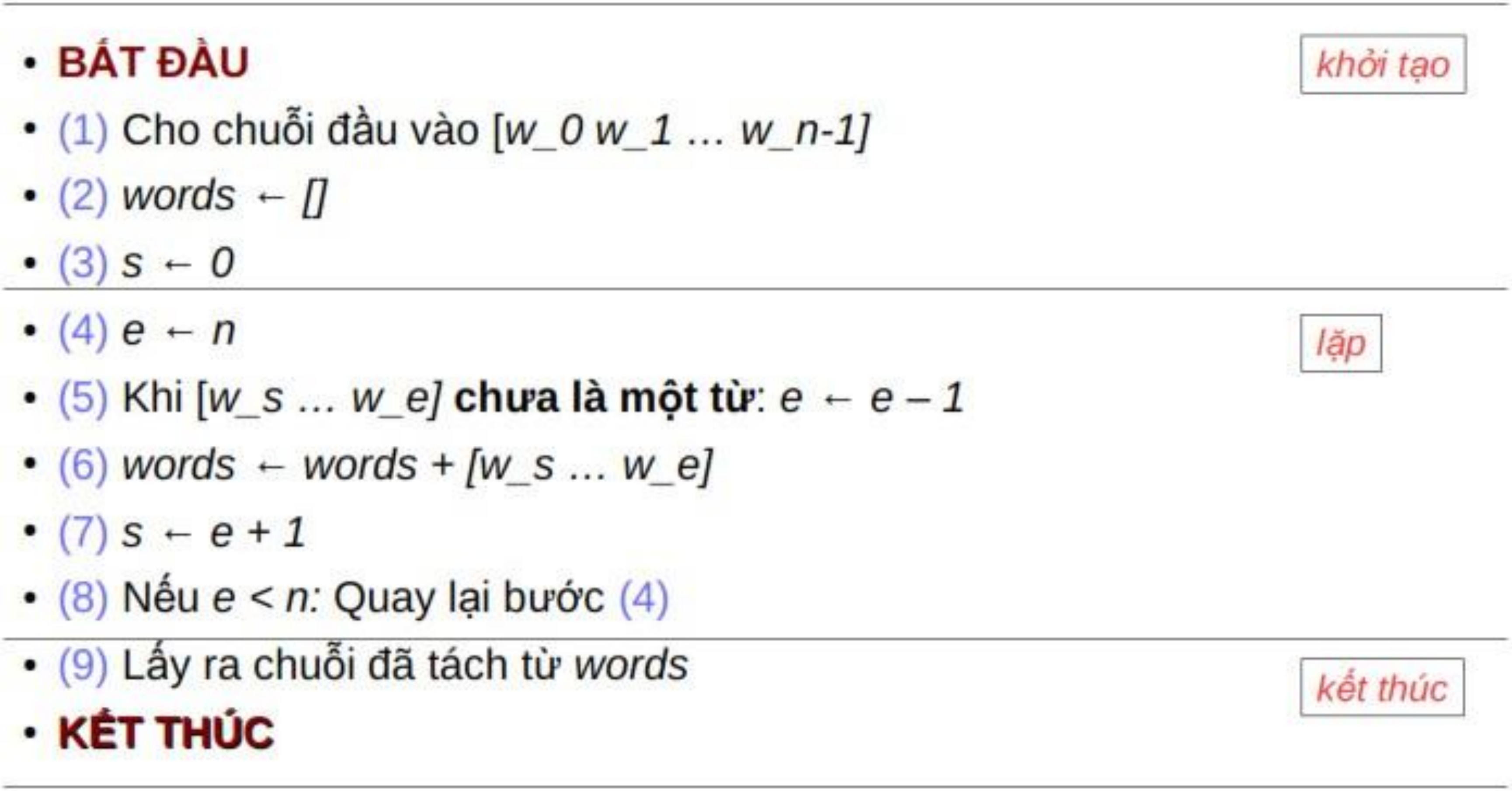
- Dictionary-based approach
- Machine learning-based approach
- A combination of these methods

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



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

khởi tạo

- (4) *e* ← *n*
- (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)

lặp

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

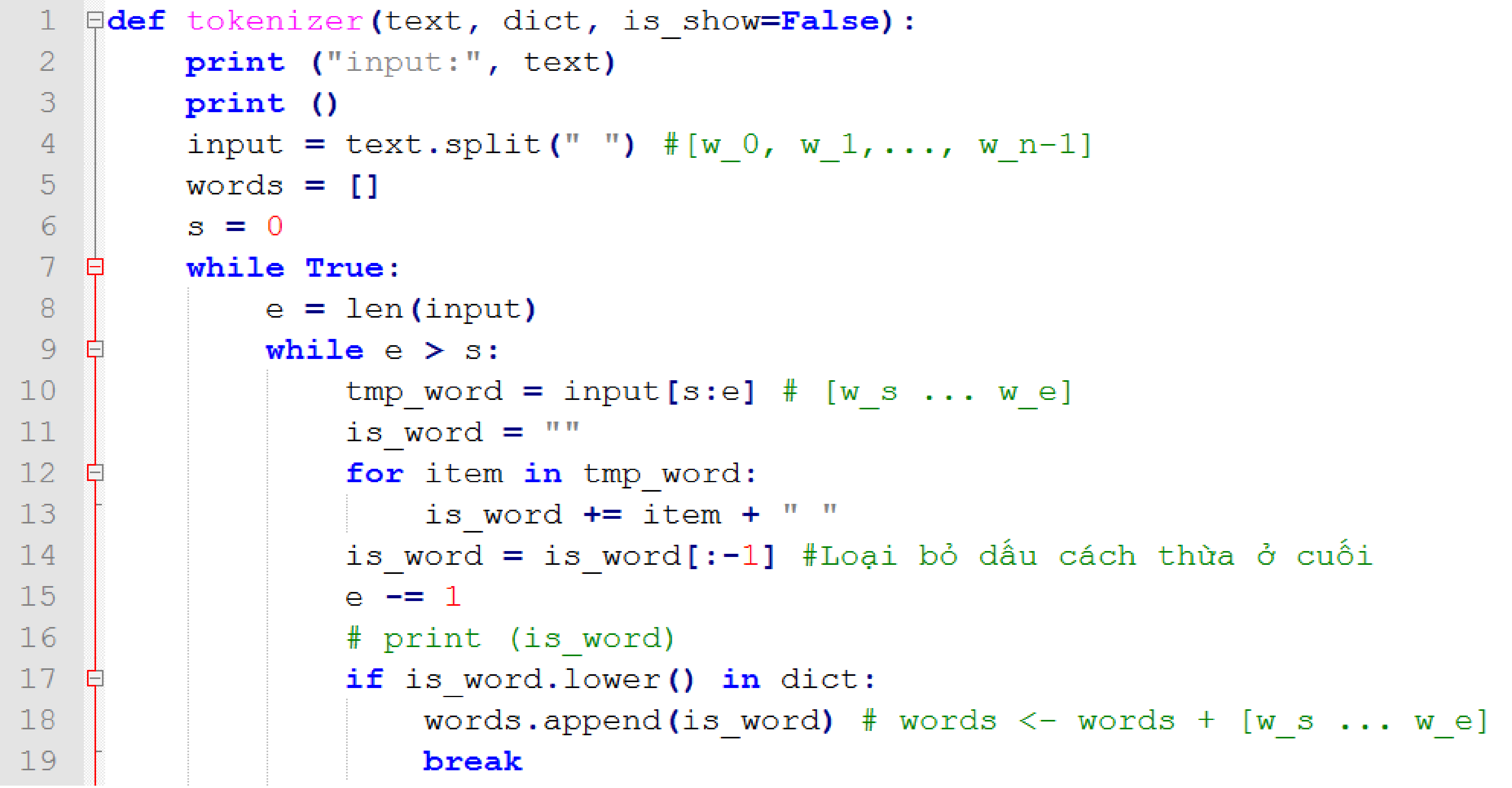
kết thúc

Longest matching algorithm

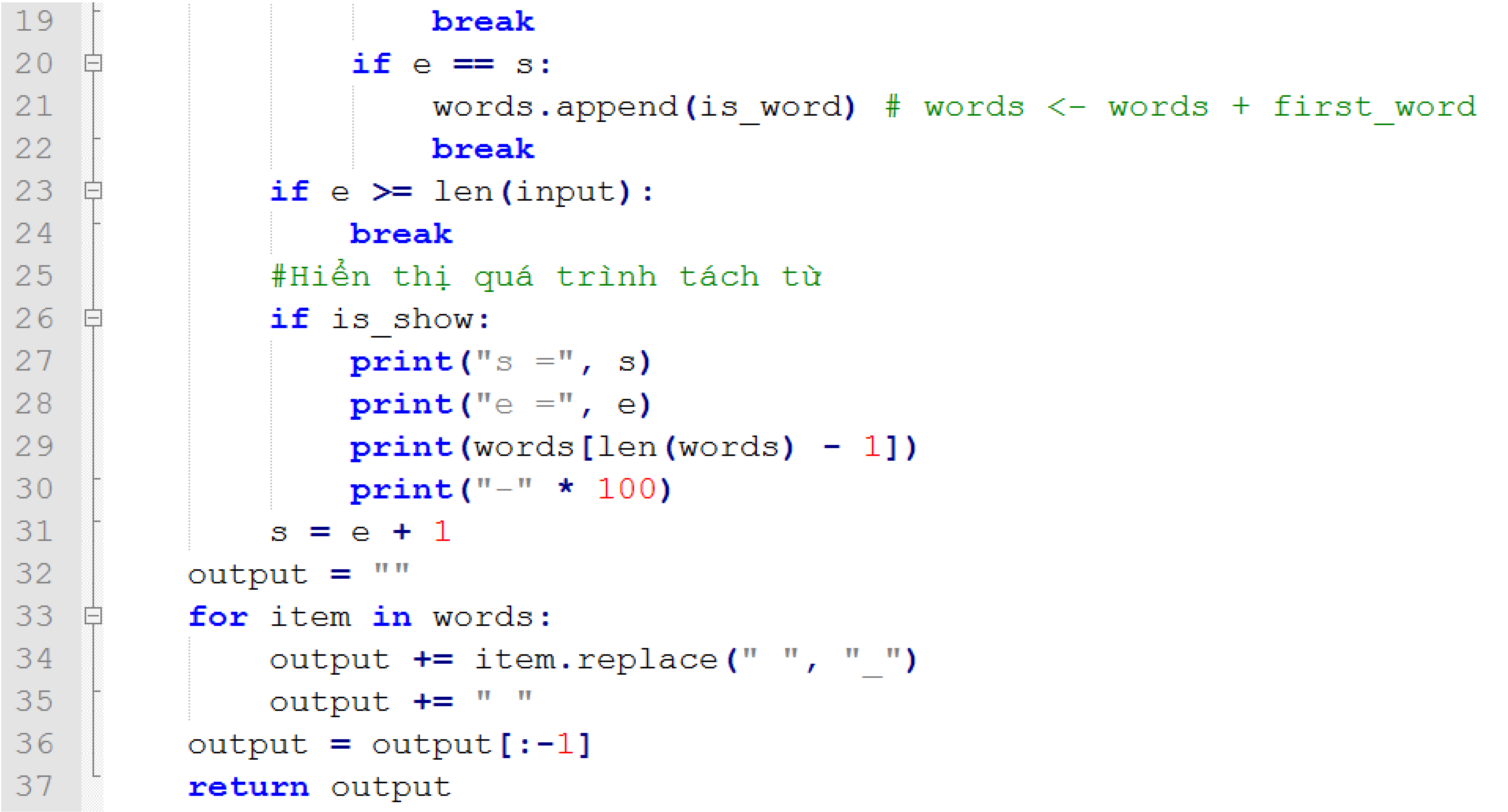
- Advantages:
  - Simple to implement
  - Reasonable complexity
  - No training data required
- Disadvantages:
  - Depend on the dictionary
  - The ambiguity issue has not been resolved

Exercices

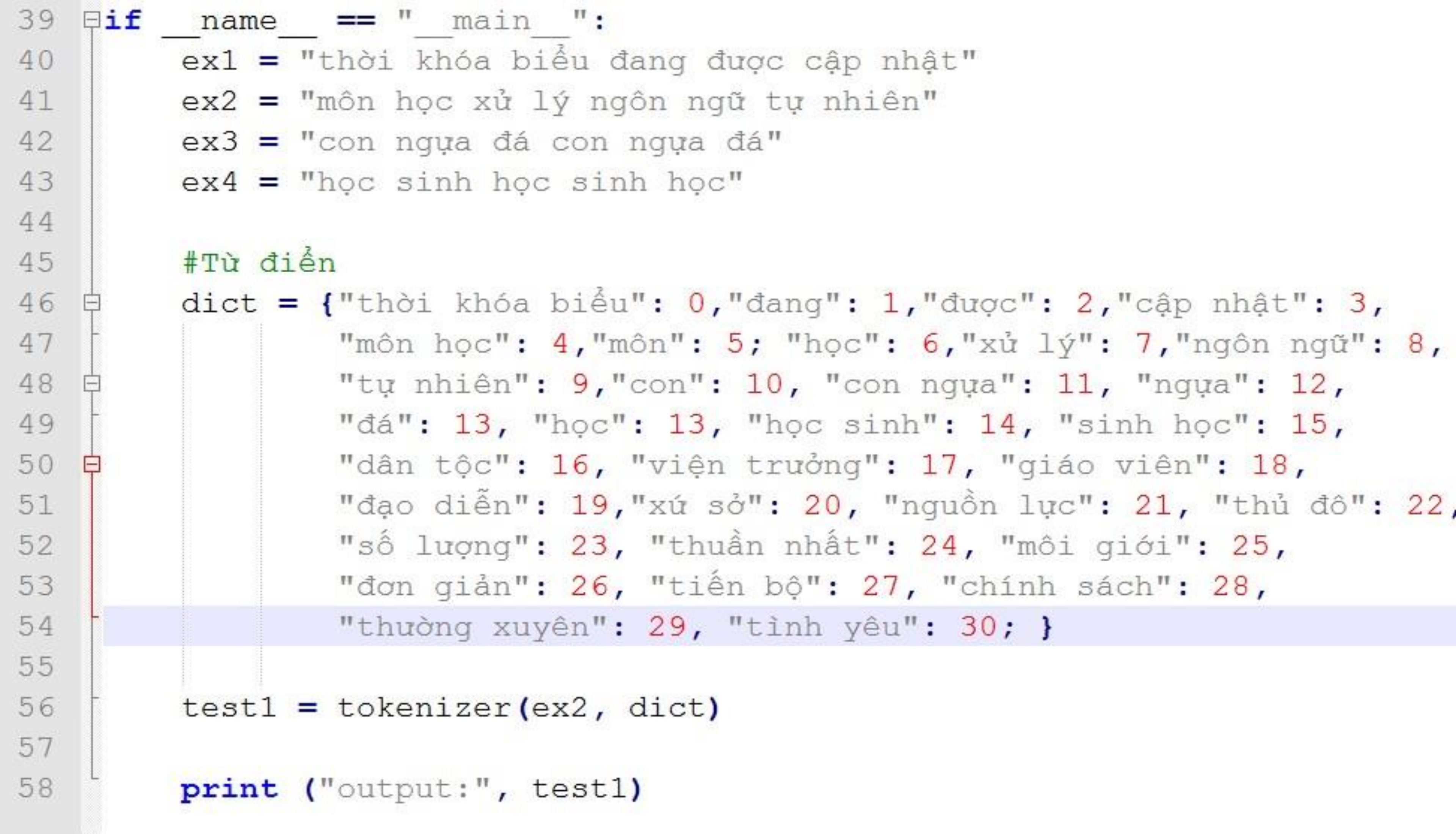
- Implement the longest word matching algorithm on Python
- Some test samples:
  - *Thời khóa biểu đang được cập nhật*
  - *Môn học xử lý ngôn ngữ tự nhiên*
  - *Ông già đi nhanh quá*
  - *Con ngựa đá con ngựa đá*
  - *Học sinh học sinh học*



```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
```



```
                break
            if e == s:
                words.append(is_word) # words <- words + first_word
                break
        if e >= len(input):
            break
        #Hiển thị quá trình tách từ
        if is_show:
            print("s =", s)
            print("e =", e)
            print(words[len(words) - 1])
            print("-" * 100)
        s = e + 1
    output = ""
    for item in words:
        output += item.replace(" ", "_")
        output += " "
    output = output[:-1]
    return output
```



```
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,"xú sở": 20, "nguồn lực": 21, "thủ đô": 22,
            "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)
```

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.

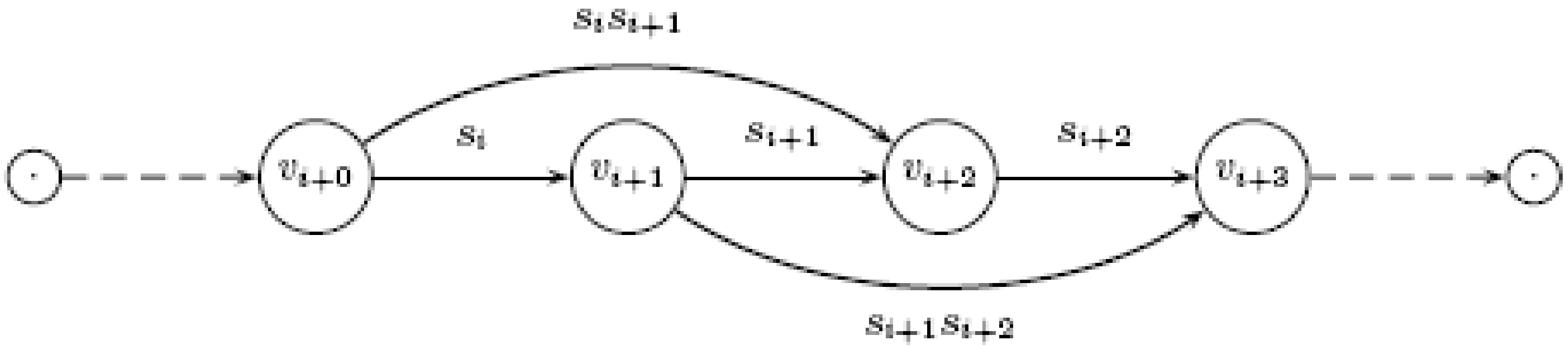
➢ Solution: list all and have a strategy to choose the best solution

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

Choosing the best solution

- Representing the input string by a sequence of syllables s_1 s_2 … 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 $G = (V,E)$, $V = \{v_0,\ v_1,\ . . . ,\ v_n,\ v_{n+1}\}$
- If syllables s_{i+1}, s_{i+2}, . . . , 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 $s_1s_2 \ldots s_n$

```
$V \leftarrow \emptyset$;
for $i = 0$ to $n + 1$ do
    $V \leftarrow V \cup \{v_i\}$;
end for
for $i = 0$ to $n$ do
    for $j = i$ to $n$ do
        if (accept($A_W$, $s_i \cdots s_j$)) then
            $E \leftarrow E \cup \{(v_i, v_{j+1})\}$;
        end if
    end for
end for
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

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

with $\lambda_1 + \lambda_2 = 1$ and $\lambda_1, \lambda_2 \geq 0$

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

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

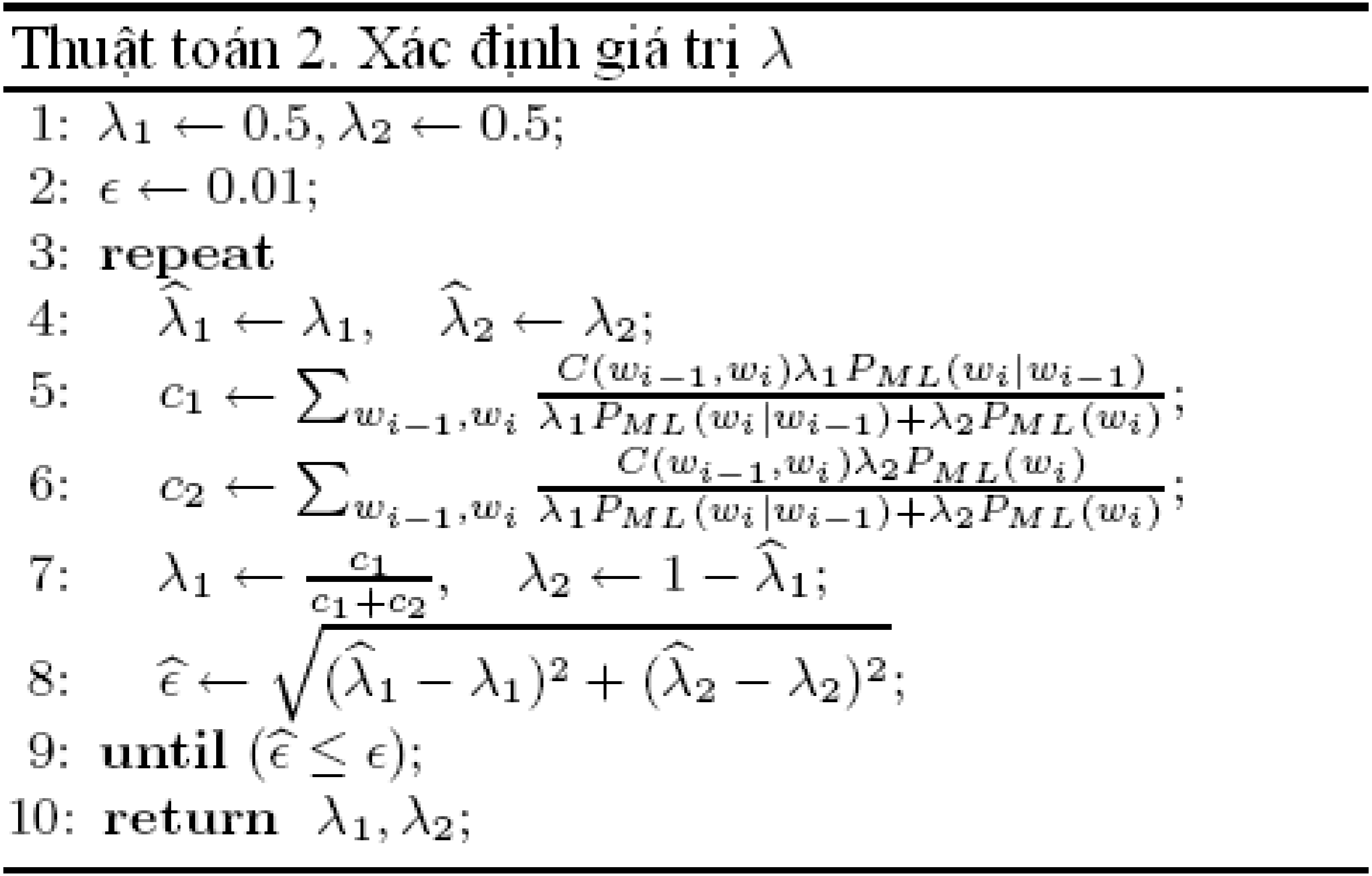
$$P(T) = \prod_{i=1}^{n} P(s_i)$$

Compute λ_1, λ_2

Choose λ_1 λ_2 to maximize:

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

Algorithm



Thuật toán 2. Xác định giá trị λ

```
$\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0.5$;
$\epsilon \leftarrow 0.01$;
repeat
    $\hat{\lambda}_1 \leftarrow \lambda_1, \quad \hat{\lambda}_2 \leftarrow \lambda_2$;
    $c_1 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_1 P_{ML}(w_i|w_{i-1})}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)}$;
    $c_2 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_2 P_{ML}(w_i)}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)}$;
    $\lambda_1 \leftarrow \frac{c_1}{c_1+c_2}, \quad \lambda_2 \leftarrow 1 - \hat{\lambda}_1$;
    $\hat{\epsilon} \leftarrow \sqrt{(\hat{\lambda}_1 - \lambda_1)^2 + (\hat{\lambda}_2 - \lambda_2)^2}$;
until $(\hat{\epsilon} \leq \epsilon)$;
return $\lambda_1, \lambda_2$;
```

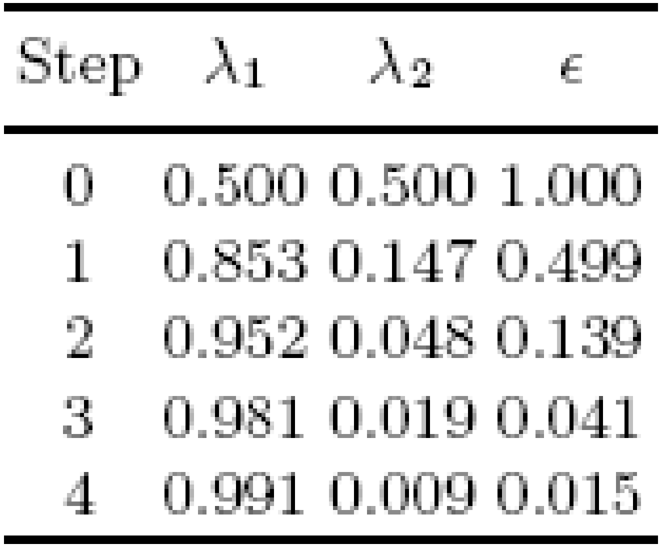
Hybrid approach

<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 ε = 0.03, the values of λ converge after 4 loops



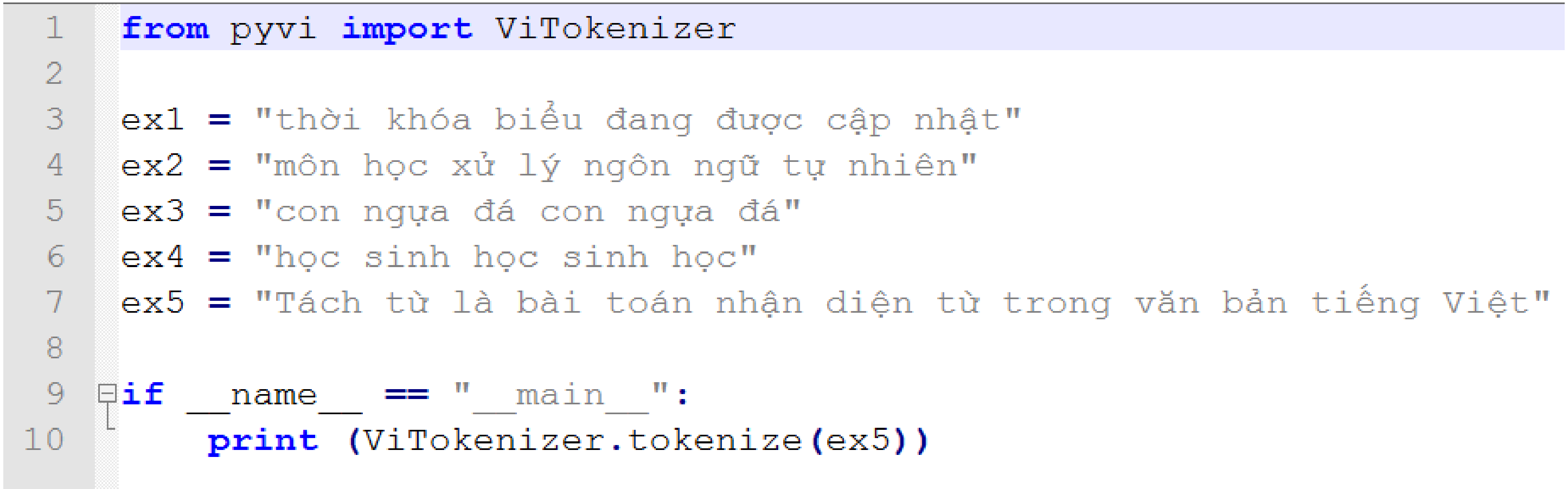
Step	λ_1	λ_2	ϵ
0	0.500	0.500	1.000
1	0.853	0.147	0.499
2	0.952	0.048	0.139
3	0.981	0.019	0.041
4	0.991	0.009	0.015

- 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))
```

Ông giả 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

IsCapitalize
IsNumber
…..
Ông già đi nhanh quá
0 1 1 1