Xử lý ngôn ngữ tự nhiên

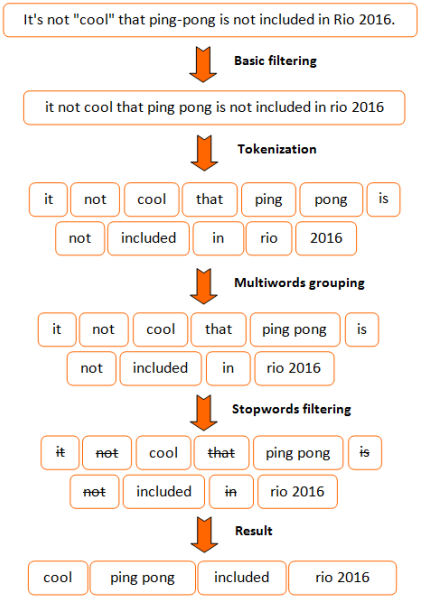
Chương 2: Xử lý và hiểu văn bản

Khoa CNTT, Đại học Kỹ thuật - Công nghệ Cần Thơ

Lưu hành nội bộ

Nội dung

- Mọi bài toán xử lý trên dữ liệu văn bản đều thuộc phạm vi ứng dụng của xử lý ngôn ngữ tự nhiên.
- Search engines: Mọi công cụ tìm kiếm đều (phần lớn) bắt đầu bằng xử lý dữ liệu người dùng nhập vào.
- Social website feeds: Giải thuật NLP xử lý các quảng cáo và posts liên quan đến dữ liệu văn bản người dùng thường sử dụng.


It's not "cool" that ping-pong is not included in Rio 2016.
Basic filtering
it not cool that ping pong is not included in rio 2016
Tokenization
it not cool that ping pong is not included in rio 2016
Multiwords grouping
it not cool that ping pong is not included in rio 2016
Stopwords filtering
it not cool that ping pong is not included in rio 2016
Result
cool ping pong included rio 2016

Tokens, tokenization

- **Tokens** là những thành phần văn bản tối giản và độc lập, tuy nhiên vẫn đảm bảo cấu trúc và ngữ nghĩa
- Một đoạn văn bản có nhiều thành phần bao gồm những câu được chia ra thành các mệnh đề, ngữ, từ.
- Đoạn văn bản có thể được chia theo câu (sentence tokenization) hoặc chia theo từ (word tokenization).
- **Tokenization** là tiến trình chia nhỏ văn bản thành các tokens.

Sentence tokenization

- Chia đoạn văn bản thành các câu (sentence segmentation).
- Chia đoạn văn bản theo ý nghĩa trọn vẹn từng câu.
- Kỹ thuật tách sử dụng các giới hạn đặc biệt (special delimiter) phân định các câu với nhau:
  - dấu chấm (.), dấu chấm phẩy (;), dấu chấm than (!)
  - ký tự xuống dòng (\n)
  - Kết hợp nhiều delimiter

Ví dụ sentence tokenization (1)

```
1 import nltk
2 sample_text = "We will discuss briefly about the basic syntax,
    structure and design philosophies. There is a defined hierarchical
    syntax for Python code which you should remember when writing code!
     Python is a really powerful programming language!"
3
4 print("Total characters: " + str(len(sample_text))) #232
```

Ví dụ sentence tokenization (2)

```
# import sentence tokenization
default_st = nltk.sent_tokenize

# apply on the sample_text
sample_sentences = default_st(text=sample_text)

print("Total sentences in sample_text: " + str(len(sample_sentences)))
for i in range(len(sample_sentences)):
    print(" sentence " + str(i + 1) + ": " + sample_sentences[i])

#Total sentences in sample_text: 3
#sentence 1: We will discuss briefly about the basic syntax, structure
    and design philosophies.
#sentence 2: There is a defined hierarchical syntax for Python code
    which you should remember when writing code!
#sentence 3: Python is a really powerful programming language!
```

Ví dụ sentence tokenization (3)

- Sử dụng text corpus được cung cấp sẵn trong thư viện NLTK:
  - `import nltk`
  - `from nltk.corpus import gutenberg`

Ví dụ sentence tokenization (4)

```
import nltk
from nltk.corpus import gutenberg

alice = gutenberg.raw(fileids="carroll-alice.txt")
print ("Total characters: " + str(len(alice)))
print("First 100 characters in the corpus\n")
print(alice[0:100])

#Total characters: 144395
#First 100 characters in the corpus
#
#[Alice's Adventures in Wonderland by Lewis Carroll 1865]
#
#CHAPTER I. Down the Rabbit-Hole
#
#Alice was
```

Ví dụ sentence tokenization (5)

```
17 # import sentence tokenization
18 default_st = nltk.sent_tokenize
19 # apply on the sample_text
20 alice_sentences = default_st(text=alice)
21
22 print("Total sentences in alice: " + str(len(alice_sentences)))
23 print("First 2 sentences in alice: ")
24 print("sentence 1: " + alice_sentences[0])
25 print("sentence 2: " + alice_sentences[1])
```

Ví dụ sentence tokenization (6)

```
26 #Total sentences in alice: 1625
27 #First 2 sentences in alice:
28 #sentence 1: [Alice's Adventures in Wonderland by Lewis Carroll 1865]
29 #
30 #CHAPTER I.
31 #sentence 2: Down the Rabbit-Hole
32 #
33 #Alice was beginning to get very tired of sitting by her sister on the
34 #bank, and of having nothing to do: once or twice she had peeped into
       the
35 #book her sister was reading, but it had no pictures or conversations
       in
36 #it, 'and what is the use of a book,' thought Alice 'without pictures
       or
37 #conversation?'
```

Word tokenization

- Chia đoạn văn bản thành các từ (word)
- Kỹ thuật quan trọng trong xử lý văn bản
- Là nền cho nhiều tiến trình sau đó: làm sạch văn bản (cleaning text), chuẩn hóa văn bản (normalize text)

Ví dụ word tokenization (1)

```
1  import nltk
2  sample_text = "the brown fox wasn't that quick and he couldn't win the
       race."
3
4  default_wt = nltk.word_tokenize
5  words = default_wt(text=sample_text)
6
7  print(len(words))
8  print(words)
9
10 #15
11 #['the', 'brown', 'fox', 'was', "n't", 'that', 'quick', 'and', 'he', '
       could', "n't", 'win', 'the', 'race', '.']
```

Ví dụ word tokenization (2)

```
whitespace_wt = nltk.WhitespaceTokenizer()
words = whitespace_wt.tokenize(sample_text)

print(len(words))
print(words)

#12
#['the', 'brown', 'fox', "wasn't", 'that', 'quick', 'and', 'he', "
    couldn't", 'win', 'the', 'race.']
```

Chuẩn hóa văn bản - Text normalization

- Chuẩn hóa văn bản là quá trình làm sạch và chuẩn văn bản để làm đầu vào cho các hệ thống xử lý ngôn ngữ tự nhiên.
- Tokenization là một công đoạn trong quá trình chuẩn hóa văn bản.
- Làm sạch văn bản (cleaning text), điều chỉnh in hoa (case conversion), chỉnh chính tả (correcting spellings), loại bỏ các từ không mang lại nhiều thông tin ý nghĩa (stopwords)
- Stemming, lemmatization

Ví dụ text normalization

```
import nltk
sample_text = "The brown fox wasn't that quick and he couldn't win the
    race. Hey that's a great deal! I just bought a phone for $199.
    @@You'll (learn) a **lot** in the BOOK. Python is an amazing
    language!@@"

default_wt = nltk.word_tokenize
words = default_wt(text=sample_text)
print(words)

#['The', 'brown', 'fox', 'was', "n't", 'that', 'quick', 'and', 'he', '
    could', "n't", 'win', 'the', 'race', '.', 'Hey', 'that', "'s", 'a',
     'great', 'deal', '!', 'I', 'just', 'bought', 'a', 'phone', 'for',
    '$', '199', '.', '@', '@', 'You', "'ll", '(', 'learn', ')', 'a',
    '**lot**', 'in', 'the', 'BOOK', '.', 'Python', 'is', 'an', 'amazing
    ', 'language', '!', '@', '@']
```

Ví dụ text normalization (2)

- Bỏ các ký tự dấu, ký tự đặc biệt.
- Xây dựng hàm để sử dụng về sau.

```
def remove_characters_after_tokenization(tokens):
    import re
    import string
    pattern = re.compile("[{}]".format(re.escape(string.punctuation)))
    filtered_tokens = list(filter(None, [pattern.sub("",token) for token
     in tokens]))
    return filtered_tokens
```

Ví dụ text normalization (3)

```
7
8  filter_list_1 = remove_characters_after_tokenization(words)
9  print(filter_list_1)
10
11 #['The', 'brown', 'fox', 'was', 'nt', 'that', 'quick', 'and', 'he', '
       could', 'nt', 'win', 'the', 'race', 'Hey', 'that', 's', 'a', 'great
       ', 'deal', 'I', 'just', 'bought', 'a', 'phone', 'for', '199', 'You
       ', 'll', 'learn', 'a', 'lot', 'in', 'the', 'BOOK', 'Python', 'is',
       'an', 'amazing', 'language']
```

```
1  #['The', 'brown', 'fox', 'was', "n't", 'that', 'quick', 'and', 'he', '
       could', "n't", 'win', 'the', 'race', '.', 'Hey', 'that', "'s", 'a',
        'great', 'deal', '!', 'I', 'just', 'bought', 'a', 'phone', 'for',
       '$', '199', '.', '@', '@', 'You', "'ll", '(', 'learn', ')', 'a',
       '**lot**', 'in', 'the', 'BOOK', '.', 'Python', 'is', 'an', 'amazing
       ', 'language', '!', '@', '@']
```

Case conversions

- Chữ in thường (lowercase)
- Chữ in hoa (uppercase)
- Case conversions chuyển đổi dữ liệu văn bản giữa 2 dạng thể hiện nêu trên.

Ví dụ Case conversions

```
1 sample_text = "The brown FOX wasn't that quick and he couldn't win the
    RACE."
2 # lowercase
3 print(sample_text.lower())
4
5 #uppercase
6 print(sample_text.upper())
7
8 #the brown fox wasn't that quick and he couldn't win the race.
9 #THE BROWN FOX WASN'T THAT QUICK AND HE COULDN'T WIN THE RACE.
```

Loại bỏ stopwords

- Stopwords là những chữ không mang lại ý nghĩa nhiều cho đoạn văn bản.
- Mỗi ngôn ngữ có danh sách stopwords chuẩn.
- Người dùng có thể tự định nghĩa stopwords riêng.

Loại bỏ stopwords

```
import nltk
english_stopwords = nltk.corpus.stopwords.words("english")
print(len(english_stopwords))
print(english_stopwords)

#153
#['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', '
    your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', '
    himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', '
    they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', '
    who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
    'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having
    ', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
    if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', '
    for', 'with', 'about', 'against', 'between', 'into', 'through', '
    during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', '
```

Ví dụ loại bỏ stopwords (1)

Sử dụng biến `filter_list_1` ở ví dụ text normalization.

```
#['The', 'brown', 'fox', 'was', 'nt', 'that', 'quick', 'and', 'he', '
    could', 'nt', 'win', 'the', 'race', 'Hey', 'that', 's', 'a', 'great
    ', 'deal', 'I', 'just', 'bought', 'a', 'phone', 'for', '199', 'You
    ', 'll', 'learn', 'a', 'lot', 'in', 'the', 'BOOK', 'Python', 'is',
    'an', 'amazing', 'language']
```

Xây dựng hàm bỏ stopwords:

```
def remove_stopwords(tokens, language="english"):
    stopword_list = nltk.corpus.stopwords.words(language)
    filter_tokens = [token for token in tokens if token not in
    stopword_list]
    return filter_tokens
```

Ví dụ loại bỏ stopwords (2)

```
filter_list_2 = remove_stopwords(filter_list_1)
print(filter_list_2)

print(filter_list_2)

#['The', 'brown', 'fox', 'nt', 'quick', 'could', 'nt', 'win', 'race', '
    Hey', 'great', 'deal', 'I', 'bought', 'phone', '199', 'You', 'learn
    ', 'lot', 'BOOK', 'Python', 'amazing', 'language']
```

Bỏ những ký tự lặp lại - repeated characters (1)

- Ví dụ: finallllyyyy
- Xây dựng hàm bỏ ký tự lặp:

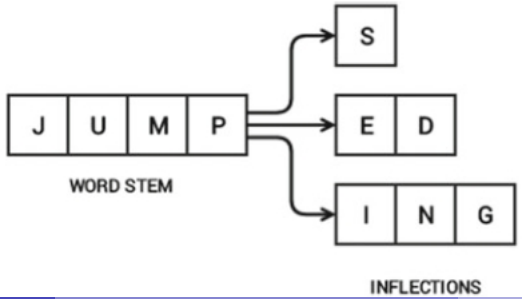
```
def remove_repeated_characters(tokens):
    from nltk.corpus import wordnet
    import re
    repeat_pattern = re.compile(r'(\w*)(\w)\2(\w*)')
    match_substitution = r'\1\2\3'
    def replace(old_word):
        if wordnet.synsets(old_word):
            return old_word
        new_word = repeat_pattern.sub(match_substitution, old_word)
        return replace(new_word) if new_word != old_word else new_word

    correct_tokens = [replace(word) for word in tokens]
    return correct_tokens
```

Repeated characters (2)

```
14
15 import nltk
16 sample_text = "My schooool is reallyyyyy ammaaazinggg"
17
18 default_wt = nltk.word_tokenize
19 tokens = default_wt(text=sample_text)
20 print(tokens)
21
22 sample_tokens = remove_repeated_characters(tokens)
23 print(sample_tokens)
24
25 #['My', 'schooool', 'is', 'reallyyyyy', 'ammaaazinggg']
26 #['My', 'school', 'is', 'really', 'amazing']
```

Stemming

- Trong ngôn ngữ học, phần đơn vị độc lập nhỏ nhất có ý nghĩa được gọi là Morphemes (hình vị)
- Morphemes bao gồm nguyên tố (stems) (còn được gọi là base - gốc) và phụ tố (affixes).



Ví dụ stemming

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()

print(ps.stem("jumping"))
print(ps.stem("jumps"))
print(ps.stem("jumped"))

#jump
#jump
#jump
```

Parts of speech (POS) tagging (1)

- Parts of speech (POS) tagging là loại từ vựng đặc biệt dùng để gán từ loại cho một token tùy theo ngữ nghĩa
  - noun, adjective, verb, adverb

Parts of speech (POS) tagging (2)

```
sentence = 'The brown fox is quick and he is jumping over the lazy dog'

# recommended tagger based on PTB
import nltk
tokens = nltk.word_tokenize(sentence)
tagged_sent = nltk.pos_tag(tokens, tagset='universal')
print(tagged_sent)

#[('The', 'DET'), ('brown', 'ADJ'), ('fox', 'NOUN'), ('is', 'VERB'), ('
    quick', 'ADJ'), ('and', 'CONJ'), ('he', 'PRON'), ('is', 'VERB'), ('
    jumping', 'VERB'), ('over', 'ADP'), ('the', 'DET'), ('lazy', 'ADJ')
    , ('dog', 'NOUN')]
```