Xử lý ngôn ngữ tự nhiên

Chương 4: Thư viện NLTK, Wordnet

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

Lưu hành nội bộ

Các tác vụ xử lý text cơ bản

1. Tokenizing văn bản thành câu
2. Tokenizing văn bản thành chữ
3. Loại bỏ stopwords trong câu đã được tokenized
4. Tìm synsets của 1 từ trong WordNet
5. Tìm kiếm ngữ nguyên (lemma), đồng nghĩa (synonyms), trái nghĩa (autonyms)
6. Tính toán sự giống nhau của từ
7. Khám phá sự kết hợp (collocations)
8. Stemming, lemmatization, Part-of-speech

Tìm synsets của một từ trong WordNet

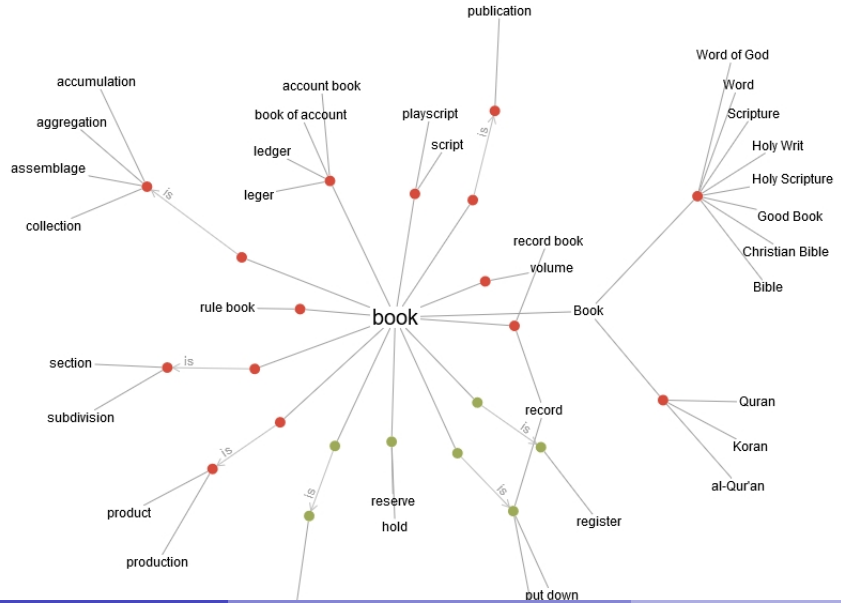
- WordNet là cơ sở dữ liệu từ vựng học của ngôn ngữ tiếng anh
- WordNet là từ điển thiết kế cụ thể cho NLP
- WordNet cung cấp giao diện đơn giản để tìm kiếm, tra từ
- Kết quả trả về là một danh sách (list) các Synset
- Synsets được sắp xếp theo cáu trúc tương tự như một cây thừa kế (inheritance tree)

Tìm synsets của một từ trong WordNet

```
from nltk.corpus import wordnet
syn = wordnet.synsets('cookbook')[0]
print(syn.name())
print(syn.definition())

#cookbook.n.01
#a book of recipes and cooking directions
```

```python
from nltk.corpus import wordnet
syns = wordnet.synsets('book')
print(len(syns))
for syn in syns:
    print(syn.name())
    print(syn.definition())

#15
#book.n.01
#a written work or composition that has been published (printed on pages bound together)
#book.n.02
#physical objects consisting of a number of pages bound together
#record.n.05
#a compilation of the known facts regarding something or someone
#script.n.01
#a written version of a play or other dramatic composition; used in preparing for a performance
# ...
```

```python
from nltk.corpus import wordnet
syns = wordnet.synsets('book')
print(len(syns))
for syn in syns:
    print(syn.examples())

#15
#['I am reading a good book on economics']
#
#['he used a large book as a doorstop']
#
#["Al Smith used to say, 'Let's look at the record'", 'his name is in all the record books']
#
#[]
#...
```


publication
accumulation
aggregation
assemblage
collection
is
account book
book of account
ledger
leger
playscript
script
is
Word of God
Word
Scripture
Holy Writ
Holy Scripture
Good Book
Christian Bible
Bible
record book
volume
rule book
book
Book
section
is
subdivision
Quran
Koran
al-Qur'an
record
is
register
is
product
production
is
reserve
hold
is
put down

Hypernyms

- Do synsets được cấu trúc dạng cây thừa kế
  - nút cha được gọi là hypernyms: trừu tượng
  - nút con được gọi là hyponyms: cụ thể
  - các nút con có ý nghĩa gần tương tự nhau
- Synsets có thể truy ngược lên nút root

Hypernyms, hyponyms

```
from nltk.corpus import wordnet
syn = wordnet.synsets('cookbook')[0]

print(syn.hypernyms())
print()
print(syn.hypernyms()[0].hyponyms())
print()
print(syn.root_hypernyms())
print()
print(syn.hypernym_paths())

[Synset('reference_book.n.01')]
```

```
13
14 [Synset('annual.n.02'), Synset('atlas.n.02'), Synset('cookbook.n.01'),
       Synset('directory.n.01'), Synset('encyclopedia.n.01'), Synset('
       handbook.n.01'), Synset('instruction_book.n.01'), Synset('
       source_book.n.01'), Synset('wordbook.n.01')]
15
16 [Synset('entity.n.01')]
17
18 [[Synset('entity.n.01'), Synset('physical_entity.n.01'), Synset('object
       .n.01'), Synset('whole.n.02'), Synset('artifact.n.01'), Synset('
       creation.n.02'), Synset('product.n.02'), Synset('work.n.02'),
       Synset('publication.n.01'), Synset('book.n.01'), Synset('
       reference_book.n.01'), Synset('cookbook.n.01')]]
```

Part of speech (POS)

- Một synsets của một từ là một list
- Trong list trả về trên, có nhiều từ loại như danh từ (n), tính từ (a), trạng từ (r), động từ (v)

Part of speech (POS)

```
from nltk.corpus import wordnet
syn = wordnet.synsets('great')[0]

print(syn.pos())
print(len(wordnet.synsets('great')))
print(len(wordnet.synsets('great', pos='n')))
print(len(wordnet.synsets('great', pos='a')))

#n
#7
#1
#6
```

Lemmas

- Lemma: dạng chuẩn tắt (canonical form) hoặ dạng ngữ nguyên (morphological form) của từ (word)

```python
from nltk.corpus import wordnet
syn = wordnet.synsets('cookbook')[0]

lemmas = syn.lemmas()
print(len(lemmas))
print(lemmas[0].name())
print(lemmas[1].name())
print(lemmas[0].synset() == lemmas[1].synset())

lems = [lemma.name() for lemma in syn.lemmas()]
print(lems)

#2
#cookbook
#cookery_book
#True
#['cookbook', 'cookery_book']
```

Lemmas

- Mỗi Synsets là một danh sách (list), mỗi element trong list có một hoặc nhiều lemmas
- Tìm tất cả các lemmas của Synsets
- Sẽ có nhiều lemmas trùng nhau, loại bỏ bằng cách đổi kiểu list thành kiểu tập hợp (set)

Lemmas

```
from nltk.corpus import wordnet

synonyms = []
for syn in wordnet.synsets('book'):
    for lemma in syn.lemmas():
        synonyms.append(lemma.name())

print(len(synonyms))
print(len(set(synonyms)))
print(synonyms)
#38
#25
#['book', 'book', 'volume', 'record', 'record_book', 'book', 'script',
    'book', 'playscript', 'ledger', 'leger', #'account_book', '
    book_of_account', 'book', 'book', 'book', 'rule_book', 'Koran', '
    Quran', "al-Qur'an", 'Book', #'Bible', 'Christian_Bible', 'Book', '
```

Autonyms - từ khác nghĩa

```
from nltk.corpus import wordnet

antonyms = []

for syn in wordnet.synsets("small"):
    for l in syn.lemmas():
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())

print(antonyms)
print(set(antonyms))

#['large', 'big', 'big']
#{'big', 'large'}
```

synonyms - từ đồng nghĩa

```
1  from nltk.corpus import wordnet
2
3  synonyms = []
4
5  for syn in wordnet.synsets('small'):
6      for lemma in syn.lemmas():
7          synonyms.append(lemma.name())
8
9  print(set(synonyms))
10
11 #{'modest', 'pocket-size', 'diminished', 'small-scale', 'lowly', '
       pocket-sized', 'minor', 'low', 'belittled', #'little', 'small', '
       humble', 'minuscule'}
```

Tính sự giống nhau của từ

- Do Synsets có cấu trúc cây, nên 2 synsets càng gần nhau thì càng giống nhau (similarity) về ý nghĩa

```
from nltk.corpus import wordnet
cb = wordnet.synsets('cookbook')[0]
ib = wordnet.synsets('instruction_book')[0]
print(cb.wup_similarity(ib))

#0.9166666666666666

table = wordnet.synsets('table')[0]
print(table.wup_similarity(ib))

#0.1111111111111111
```

- Synsets `table` và synsets `instruction_book` giống nhau 11%
- Kiểm tra các hypernyms chung của 2 synsets

```
12 print(sorted(table.common_hypernyms(ib)))
13
14 #[Synset('entity.n.01')]
```

Tìm collocations

- Collocations là 2 hoặc nhiều word xuất hiện gần nhau thường xuyên, ví dụ word `Cần` và word `Thơ`
- Tại ngữ cảnh khác nhau thì collocations cũng khác nhau
- Thư viện nltk hỗ trợ: BigramCollocationFinder và TrigramCollocationFinder

```
from nltk.corpus import webtext
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures
from nltk.corpus import stopwords

words = [w.lower() for w in webtext.words('singles.txt')]

stopset = set(stopwords.words('english'))
filter_stops = lambda w: len(w) < 3 or w in stopset

bcf = BigramCollocationFinder.from_words(words)

bcf.apply_word_filter(filter_stops)
# appear more than 4 times
print(bcf.nbest(BigramAssocMeasures.likelihood_ratio, 4))

#[('would', 'like'), ('age', 'open'), ('medium', 'build'), ('social', '
    drinker')]
```

Stemming words

- Kỹ thuật gởi bỏ đuôi của từ (ed, ing)
  - PorterStemmer
  - LancasterStemmer
  - SnowballStemmer
  - RegexpStemmer

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
print(stemmer.stem('cooking'))
print(stemmer.stem('cookery'))

#cook
#cookeri

from nltk.stem import LancasterStemmer
stemmer = LancasterStemmer()
print(stemmer.stem('cooking'))
print(stemmer.stem('cookery'))

#cook
#cookery
```

```
from nltk.stem import SnowballStemmer
stemmer = LancasterStemmer()
print(stemmer.stem('cooking'))
print(stemmer.stem('cookery'))

#cook
#cookery

from nltk.stem import RegexpStemmer
stemmer = RegexpStemmer('ing')
print(stemmer.stem('cooking'))
print(stemmer.stem('cookery'))
print(stemmer.stem('ingleside'))

#cook
#cookery
#leside
```

Khác nhau giữa Stemming và Lemmatization

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

print(stemmer.stem('stones'))
print(stemmer.stem('speaking'))
print(stemmer.stem('bedroom'))
print(stemmer.stem('jokes'))
print(stemmer.stem('lisa'))
print(stemmer.stem('purple'))
print('————————————————————')
```

```
14
15 print(lemmatizer.lemmatize('stones'))
16 print(lemmatizer.lemmatize('speaking'))
17 print(lemmatizer.lemmatize('bedroom'))
18 print(lemmatizer.lemmatize('jokes'))
19 print(lemmatizer.lemmatize('lisa'))
20 print(lemmatizer.lemmatize('purple'))
```

```
21
22 stone
23 speak
24 bedroom
25 joke
26 lisa
27 purpl
28 ----------
29 stone
30 speaking
31 bedroom
32 joke
33 lisa
34 purple
```

Thay thế word bằng regular expressions

- Tạo file replacers.py có nội dung như sau:

```
import re

replacement_patterns = [
    (r'won\'t', 'will not'),
    (r'can\'t', 'cannot'),
    (r'i\'m', 'i am'),
    (r'ain\'t', 'is not'),
    (r'(\w+)\'ll', '\g<1> will'),
    (r'(\w+)n\'t', '\g<1> not'),
    (r'(\w+)\'ve', '\g<1> have'),
    (r'(\w+)\'s', '\g<1> is'),
    (r'(\w+)\'re', '\g<1> are'),
    (r'(\w+)\'d', '\g<1> would'),
]
```

```
15
16 class RegexpReplacer(object):
17     def __init__(self, patterns=replacement_patterns):
18         self.patterns = [(re.compile(regex), repl) for (regex, repl) in
       patterns]
19
20     def replace(self, text):
21         s = text
22
23         for (pattern, repl) in self.patterns:
24             s = re.sub(pattern, repl, s)
25
26         return s
```

- File chương trình chính cùng cấp với file replacers.py

```
from replacers import RegexpReplacer
replacer = RegexpReplacer()
print(replacer.replace("can't is a contraction"))
print(replacer.replace("I should've done that thing I didn't do"))

#cannot is a contraction
#I should have done that thing I did not do
```

- Thay thế ký tự lặp lại (repeating characters): sinh viên xem bài cũ

Data kèm theo thư viện NLTK

- Xem lại bài trước về cách cài đặt data của thư viện NLTK
- Ví dụ data được cài vào thư mục `D:\nltk_data`
- Thư mục `D:\nltk_data\corpora` chứa các data sau khi cài đặt

```python
from nltk.corpus import names
print(names.fileids())
print(len(names.words('female.txt')))
print(len(names.words('male.txt')))

from nltk.corpus import words
print(words.fileids())
print(len(words.words('en-basic')))
print(len(words.words('en')))

#['female.txt', 'male.txt']
#5001
#2943
#['en', 'en-basic']
#850
#235886
```

Wordlist do người dùng tự tạo

- Tạo thư mục `nlp-corpus` trong `D:\nltk_data\corpora`
- Tạo file `mywords.txt`, lưu trong thư mục `nlp-corpus`, với nội dung sau:

```
1 nltk
2 corpus
3 corpora
4 wordnet
```

```
import nltk.data
from nltk.corpus.reader import WordListCorpusReader

reader = WordListCorpusReader('D:/nltk_data/corpora/nlp-corpus/', ['mywords.txt'])
print(reader.words())
print(reader.fileids())

#['nltk', 'corpus', 'corpora', 'wordnet']
#['mywords.txt']
```

Part-of-speech tagging

- Xem lại bài cũng đối với tác vụ này
- Trong ví dụ này, chúng ta sẽ load từ file đã có nội dung word/tag
- Tạo file `txt-pos.txt` trong thư mục `D:\nltk_data\corpora\nlp-corpus` đã tạo ở slides trước
- Nội dung file: `The/at-tl expense/nn and/cc time/nn involved/vbn are/ber astronomical/jj ./.`

```
from nltk.corpus.reader import TaggedCorpusReader
reader = TaggedCorpusReader("D:/nltk_data/corpora/nlp-corpus", ["txt-
    pos.txt"], tagset="en-brown")
print(reader.words())
print(reader.sents())
print(reader.paras())
print(reader.tagged_words(tagset="universal"))
print(reader.tagged_sents(tagset="universal"))
print(reader.tagged_paras(tagset="universal"))
```

```
9
10 #['The', 'expense', 'and', 'time', 'involved', 'are', ...]
11 #[['The', 'expense', 'and', 'time', 'involved', 'are', 'astronomical']]
12 #[[['The', 'expense', 'and', 'time', 'involved', 'are', 'astronomical
       ']]]
13 #[('The', None), ('expense', None), ('and', None), ...]
14 #[[('The', None), ('expense', None), ('and', None), ('time', None), ('
       involved', None), ('are', None), #('astronomical', None)]]
15 #[[[('The', None), ('expense', None), ('and', None), ('time', None), ('
       involved', None), ('are', None), #('astronomical', None)]]]
```

Sử dụng WorNet cho tác vụ tagging

- Tạo file `taggers.py` với nội dung như sau:

```
from nltk.tag import NgramTagger, SequentialBackoffTagger
from nltk.corpus import wordnet
from nltk.probability import FreqDist

class WordNetTagger(SequentialBackoffTagger):
    def __init__(self, *args, **kwargs):
        SequentialBackoffTagger.__init__(self, *args, **kwargs)
        self.wordnet_tag_map = {
        'n': 'NN',
        's': 'JJ',
        'a': 'JJ',
        'r': 'RB',
        'v': 'VB'
        }
```

```
15      def choose_tag(self, tokens, index, history):
16          word = tokens[index]
17          fd = FreqDist()
18
19          for synset in wordnet.synsets(word):
20              fd[synset.pos()] += 1
21
22          if not fd: return None
23          return self.wordnet_tag_map.get(fd.max())
```

- Tạo file chương trình chính cùng cấp với file `taggers.py`

```
from taggers import WordNetTagger
wt = WordNetTagger()
print(wt.tag(['food', 'is', 'great']))

#[('food', 'NN'), ('is', 'VB'), ('great', 'JJ')]
```