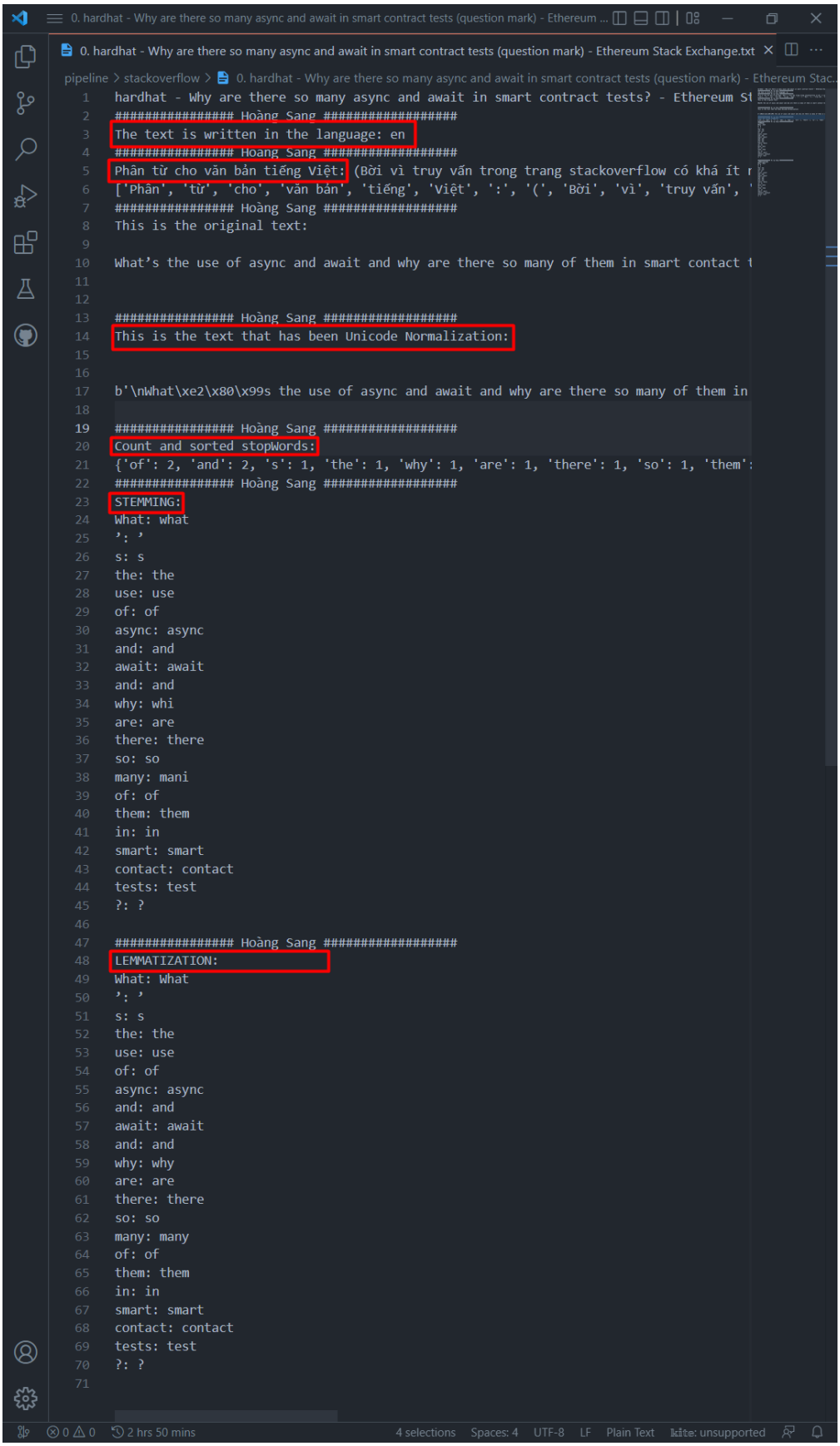
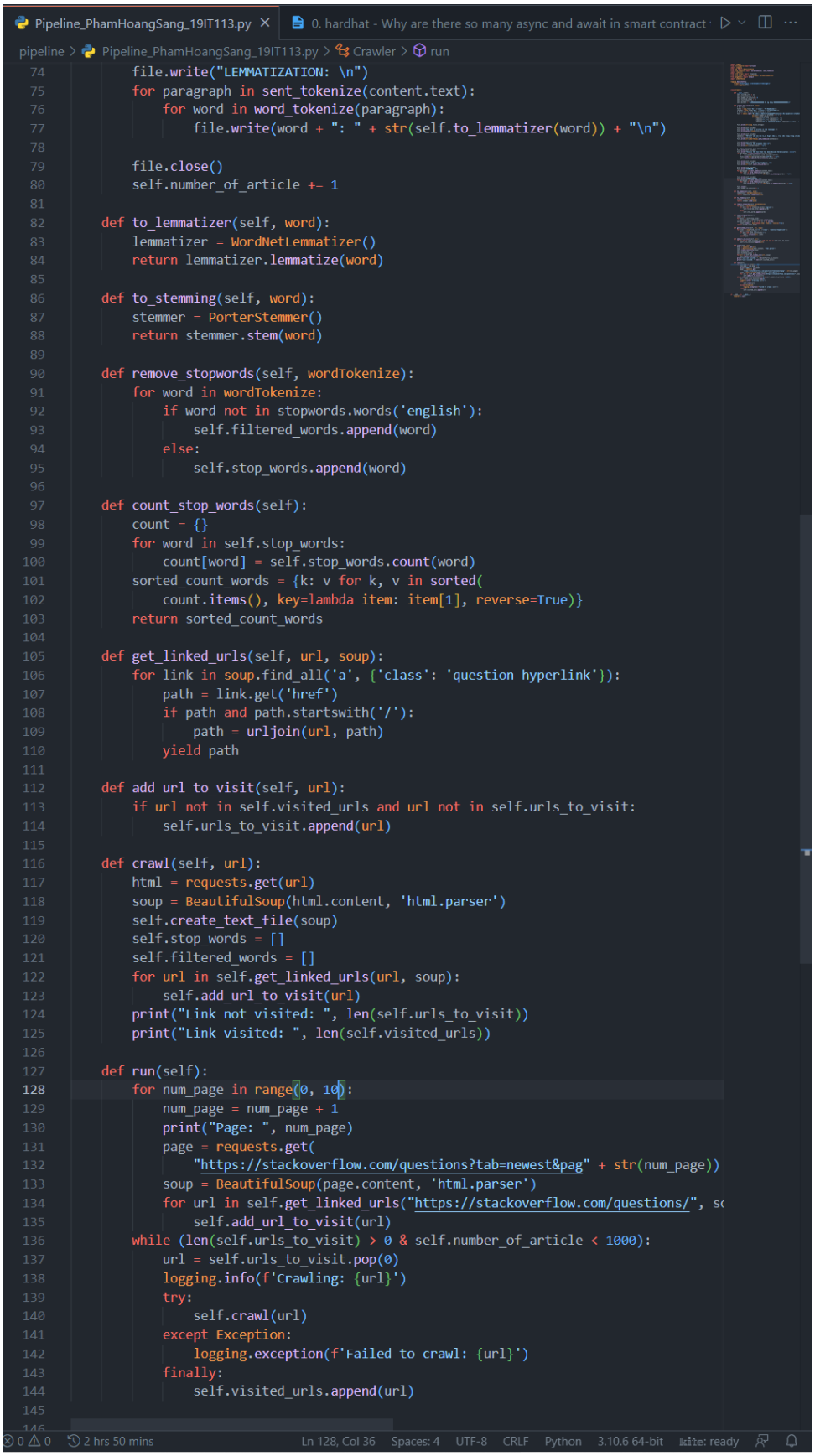
Kết quả


0. hardhat - Why are there so many async and await in smart contract tests (question mark) - Ethereum ...
0. hardhat - Why are there so many async and await in smart contract tests (question mark) - Ethereum Stack Exchange.txt
pipeline > stackoverflow > 0. hardhat - Why are there so many async and await in smart contract tests (question mark) - Ethereum Stac...
1 hardhat - Why are there so many async and await in smart contract tests? - Ethereum St
2 ################ Hoàng Sang ##################
3 The text is written in the language: en
4 ################ Hoàng Sang ##################
5 Phân từ cho văn bản tiếng Việt: (Bởi vì truy vấn trong trang stackoverflow có khá ít r
6 ['Phân', 'từ', 'cho', 'văn bản', 'tiếng', 'Việt', ':', '(', 'Bởi', 'vì', 'truy vấn',
7 ################ Hoàng Sang ##################
8 This is the original text:
9
10 What’s the use of async and await and why are there so many of them in smart contact t
11
12
13 ################ Hoàng Sang ##################
14 This is the text that has been Unicode Normalization:
15
16
17 b'\nWhat\xe2\x80\x99s the use of async and await and why are there so many of them in
18
19 ################ Hoàng Sang ##################
20 Count and sorted stopWords:
21 {'of': 2, 'and': 2, 's': 1, 'the': 1, 'why': 1, 'are': 1, 'there': 1, 'so': 1, 'them':
22 ################ Hoàng Sang ##################
23 STEMMING:
24 What: what
25 ’: ’
26 s: s
27 the: the
28 use: use
29 of: of
30 async: async
31 and: and
32 await: await
33 and: and
34 why: whi
35 are: are
36 there: there
37 so: so
38 many: mani
39 of: of
40 them: them
41 in: in
42 smart: smart
43 contact: contact
44 tests: test
45 ?: ?
46
47 ################ Hoàng Sang ##################
48 LEMMATIZATION:
49 What: What
50 ’: ’
51 s: s
52 the: the
53 use: use
54 of: of
55 async: async
56 and: and
57 await: await
58 and: and
59 why: why
60 are: are
61 there: there
62 so: so
63 many: many
64 of: of
65 them: them
66 in: in
67 smart: smart
68 contact: contact
69 tests: test
70 ?: ?
71
0 0 2 hrs 50 mins
4 selections Spaces: 4 UTF-8 LF Plain Text kite: unsupported

Code


Pipeline_PhamHoangSang_19IT113.py
0. hardhat - Why are there so many async and await in smart contract
pipeline > Pipeline_PhamHoangSang_19IT113.py > Crawler > run
import codecs
from urllib.parse import urljoin
import logging
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize, sent_tokenize
import requests
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from langdetect import detect
import underthesea
logging.basicConfig(
format='%(asctime)s %(levelname)s:%(message)s',
level=logging.INFO)
class Crawler:
def __init__(self):
self.visited_urls = []
self.urls_to_visit = []
self.number_of_article = 0
self.filtered_words = []
self.stop_words = []
self.divider = "\n################ Hoàng Sang ##################\n"
def create_text_file(self, soup):
# Get text
title = soup.find('h1', {'class': 'fs-headline1'})
content = soup.find('div', {'class': 'js-post-body'})
# Write text to file with encoding is utf-8
file = codecs.open("D:\\E23.1\\NaturalLanguageProcessing\\Ex\\pipeline\\stack
str(self.number_of_article) + ". " +
str(soup.title.string
.replace("|", "").replace("(", "")
.replace("!", "").replace("/", "")
.replace("?", " (question mark)").replace(",", "")) +
file.write(str(soup.title.string))
file.write(self.divider)
file.write("The text is written in the language: ")
file.write(detect(soup.title.string))
file.write(self.divider)
sentence = "Phân từ cho văn bản tiếng Việt: (Bởi vì truy vấn trong trang stac
file.write(sentence + "\n")
file.write(str(underthesea.word_tokenize(sentence)))
file.write(self.divider)
file.write("This is the original text:\n")
file.write(str(content.text) + "\n")
# split sentences with sent_tokenize
file.write(self.divider)
file.write("This is the text that has been Unicode Normalization: \n\n\n")
for paragraph in sent_tokenize(content.text):
# Unicode Normalization with .encode("utf-8")
file.write(str(paragraph.encode("utf-8")) + "\n")
self.remove_stopwords(word_tokenize(paragraph))
file.write(self.divider)
file.write("Count and sorted stopWords: \n")
file.write(str(self.count_stop_words()))
file.write(self.divider)
file.write("STEMMING: \n")
for paragraph in sent_tokenize(content.text):
for word in word_tokenize(paragraph):
file.write(word + ": " + str(self.to_stemming(word)) + "\n")
file.write(self.divider)
0 0 2 hrs 50 mins Ln 128, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.6 64-bit kite: ready


Pipeline_PhamHoangSang_19IT113.py
0. hardhat - Why are there so many async and await in smart contract
pipeline > Pipeline_PhamHoangSang_19IT113.py > Crawler > run
file.write("LEMMATIZATION: \n")
for paragraph in sent_tokenize(content.text):
for word in word_tokenize(paragraph):
file.write(word + ": " + str(self.to_lemmatizer(word)) + "\n")
file.close()
self.number_of_article += 1
def to_lemmatizer(self, word):
lemmatizer = WordNetLemmatizer()
return lemmatizer.lemmatize(word)
def to_stemming(self, word):
stemmer = PorterStemmer()
return stemmer.stem(word)
def remove_stopwords(self, wordTokenize):
for word in wordTokenize:
if word not in stopwords.words('english'):
self.filtered_words.append(word)
else:
self.stop_words.append(word)
def count_stop_words(self):
count = {}
for word in self.stop_words:
count[word] = self.stop_words.count(word)
sorted_count_words = {k: v for k, v in sorted(
count.items(), key=lambda item: item[1], reverse=True)}
return sorted_count_words
def get_linked_urls(self, url, soup):
for link in soup.find_all('a', {'class': 'question-hyperlink'}):
path = link.get('href')
if path and path.startswith('/'):
path = urljoin(url, path)
yield path
def add_url_to_visit(self, url):
if url not in self.visited_urls and url not in self.urls_to_visit:
self.urls_to_visit.append(url)
def crawl(self, url):
html = requests.get(url)
soup = BeautifulSoup(html.content, 'html.parser')
self.create_text_file(soup)
self.stop_words = []
self.filtered_words = []
for url in self.get_linked_urls(url, soup):
self.add_url_to_visit(url)
print("Link not visited: ", len(self.urls_to_visit))
print("Link visited: ", len(self.visited_urls))
def run(self):
for num_page in range(0, 10):
num_page = num_page + 1
print("Page: ", num_page)
page = requests.get(
"https://stackoverflow.com/questions?tab=newest&pag" + str(num_page))
soup = BeautifulSoup(page.content, 'html.parser')
for url in self.get_linked_urls("https://stackoverflow.com/questions/", s
self.add_url_to_visit(url)
while (len(self.urls_to_visit) > 0 & self.number_of_article < 1000):
url = self.urls_to_visit.pop(0)
logging.info(f'Crawling: {url}')
try:
self.crawl(url)
except Exception:
logging.exception(f'Failed to crawl: {url}')
finally:
self.visited_urls.append(url)
0 0
2 hrs 50 mins
Ln 128, Col 36
Spaces: 4
UTF-8
CRLF
Python
3.10.6 64-bit
Kite: ready