

test_wordprocessing.py

MIT License

Copyright 2023 auto_anki

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Tests the keyword_extractor method

Tests the duplicate_word_removal method

Tests the merge_slide_with_same_header method

Tests the merge_slide_with_same_slide_number method

Tests the extract_noun_chunks method

Tests the construct_search_query method

```
import unittest
import json
from wordprocessing import keyword_extractor, duplicate_word_removal, merge_slide_with_same_headers, \
    merge_slide_with_same_slide_number, extract_noun_chunks, construct_search_query

class TestWordProcessing(unittest.TestCase):

    def setUp(self) -> None:
        pass

    def test_keyword_extractor(self):

        data = [{"Header": "This is a Header", "Paragraph": "This is a Paragraph", "slide": 10}]
        keywords = keyword_extractor(data)
        data[0]["Header_keywords"] = ["header"]
        data[0]["Paragraph_keywords"] = ["paragraph"]
        self.assertEqual(keywords, data)

    def test_duplicate_word_removal(self):

        data = [{"Header": "This is a Header, and this is a Header", "Paragraph": "This is a Paragraph, and this is a "
            "Paragraph",
            "Header_keywords": ["header", "header"],
            "Paragraph_keywords": ["paragraph", "paragraph"], "slide": 10}]
        remove_duplicates = duplicate_word_removal(data)
        data[0]["Header_keywords"] = ["header"]
        data[0]["Paragraph_keywords"] = ["paragraph"]
        self.assertEqual(data, remove_duplicates)

    def test_merge_slide_with_same_header(self):

        data = [{"Header": "This is a Header", "Paragraph": "This is paragraph one", "Header_keywords": ["header"],
            "Paragraph_keywords": ["paragraph", "one"], "slide": 10},
            {"Header": "This is a Header", "paragraph": "This is paragraph two", "Header_keywords": ["header"],
            "Paragraph_keywords": ["paragraph", "two"], "slide": 11}]
        merged_data = merge_slide_with_same_headers(data)
        data[0]["slides"] = [10, 11]
        data[0]["Paragraph_keywords"] = ["paragraph", "one", "paragraph", "two"]
        data[0].pop("Paragraph", None)
        data[0].pop("slide", None)
        data.pop()
        self.assertEqual(data, merged_data)

    def test_merge_slide_with_same_slide_number(self):

        input_data = [{"Header": 'Dimensionality Reduction PCA',
            'Paragraph': 'This is paragraph 1',
            'slide': 8, 'Header_keywords': ['dimensionality', 'reduction', 'pca'],
            'Paragraph_keywords': ['dimensionality', 'reduction', 'purposes']},
            {'Header': 'Gratuitous ARP',
            'Paragraph': 'This is paragraph 2',
            'slide': 9, 'Header_keywords': ['arp'],
            'Paragraph_keywords': ['machine', 'mapping', 'arp', 'caches']},
            {'Header': 'Dimensionality Reduction PCA',
            'Paragraph': 'This is paragraph 3',
            'slide': 9, 'Header_keywords': ['dimensionality', 'reduction', 'pca'],
            'Paragraph_keywords': ['goal', 'projection']}]

        merged_data = merge_slide_with_same_slide_number(input_data)
        output_data = [{"Header": 'Dimensionality Reduction PCA',
            'slide': 8, 'Header_keywords': ['dimensionality', 'reduction', 'pca'],
            'Paragraph_keywords': ['dimensionality', 'reduction', 'purposes']},
            {'Header': 'Gratuitous ARP',
            'slide': 9, 'Header_keywords': ['arp', 'dimensionality', 'reduction', 'pca'],
            'Paragraph_keywords': ['machine', 'mapping', 'arp', 'caches', 'goal', 'projection']}]

        self.assertEqual(output_data, merged_data)

    def test_extract_noun_chunks(self):
        '''Test the extract_noun_chunks method'''

        input_data = [{"Header": "This is a noun chunk", "Paragraph": "This is a noun chunk", "slide":1}]
        noun_chunks_result = extract_noun_chunks(input_data)
        output_data = [{"Header": 'This is a noun chunk', 'Paragraph': 'This is a noun chunk', 'slide': 1, 'Header_keywords': ['noun c

        self.assertEqual(output_data, noun_chunks_result)

    def test_construct_search_query(self):
        '''Test the search query method'''

        with open("code/data/Test_3.json", mode="r") as file:
            input_data = json.load(file)
        with open("code/data/Test_3_Result.json", mode="r") as file:
            output_data = json.load(file)

        search_query = construct_search_query(input_data)
```

```
self.assertEqual(output_data, search_query)
```

```
if __name__ == "__main__":  
    unittest.main()
```