

test_integration.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.

Given some predefined PDFs, make sure that the results are not None

Given the ARP file, tests that the tools are able to extract words and form groupings

tests getting pdf -> getting groupings

check that the structure is right

assert that each slide has a header, paragraph, and slide

assert the field is present

assert the field is not None

assert the type is correct

Given the arp file, tests that the tools are able to extract words, form groupings, and extract_noun_chunks

tests getting pdf -> getting groupings

check that the structure is right

[{"Header": "", "Paragraph": "", "Header_keywords": [], "Paragraph_keywords": [], slide: int}] assert the structure is correct

assert that each slide has a header, paragraph, and slide

assert the field is not None

assert the type is correct

Given the arp file, tests that the tools are able to merge_slide_with_same_headers

tests getting pdf -> getting merge_slide_with_same_headers

check that the structure is right

[{"Header": "", "Header_keywords": [], "Paragraph_keywords": [], slides: [int]}] assert the structure is correct

assert that each slide has a header, paragraph, and slide

assert the field is present

assert the field is not None

assert the type is correct

```
from extract_sizes import extract_words, text_to_groupings
from wordprocessing import extract_noun_chunks, merge_slide_with_same_headers, duplicate_word_removal, \
    construct_search_query
```

```
PDF_NAME = "data/arp.pdf"
```

```
def test_groupings():
    """Tests that the tools are able to extract words and form groupings"""
```

```
pdf_doc = extract_words(PDF_NAME)
assert pdf_doc is not None
groupings = text_to_groupings(pdf_doc)
assert groupings is not None
```

```
structure_dict = {'Header': str, 'Paragraph': str, 'slide': int}
for slide in groupings:
```

```
    for struct in structure_dict.keys():
```

```
        assert struct in slide
```

```
        assert slide[struct] is not None
```

```
        assert type(slide[struct]) is structure_dict[struct]
```

```
def test_chunks():
```

```
pdf_doc = extract_words(PDF_NAME)
assert pdf_doc is not None
groupings = text_to_groupings(pdf_doc)
assert groupings is not None
chunks = extract_noun_chunks(groupings)
```

```
assert chunks is not None
```

```
structure_dict = {'Header': str, 'Paragraph': str, 'Header_keywords': list, 'Paragraph_keywords': list, 'slide': int}
for slide in chunks:
```

```
    for struct in structure_dict.keys():
        assert struct in slide, f'{struct} should be in slide'
```

```
        assert slide[struct] is not None, f'{struct} in slide should not be none'
```

```
        assert type(slide[struct]) is structure_dict[struct], f'{struct} should be a {structure_dict[struct}]'
```

```
def test_merge_slide_with_same_headers():
    """Tests that the tools are able to merge_slide_with_same_headers"""
```

```
pdf_doc = extract_words(PDF_NAME)
assert pdf_doc is not None
groupings = text_to_groupings(pdf_doc)
assert groupings is not None
chunks = extract_noun_chunks(groupings)
```

```
assert chunks is not None
chunks = merge_slide_with_same_headers(chunks)
assert chunks is not None
```

```
structure_dict = {'Header': str, 'Header_keywords': list, 'Paragraph_keywords': list, 'slides': list}
for slide in chunks:
```

```
    for struct in structure_dict.keys():
```

```
        assert struct in slide, f'{struct} should be in slide'
```

```
        assert slide[struct] is not None, f'{struct} in slide should not be none'
```

```
        assert type(slide[struct]) is structure_dict[struct], f'{struct} should be a {structure_dict[struct}]'
```

```
def test_duplicate_word_removal():
```

Given the arp file, tests that the tools are able to duplicate_word_removal

tests getting pdf -> getting merge_slide_with_same_headers

[{"Header": "", "Header_keywords": [], "Paragraph_keywords": [], "slides":
[int]]] assert the structure is correct

assert that each slide has a header, paragraph, and slide

assert the field is present

assert the field is not None

assert the type is correct

Given the arp file, tests that the tools are able to construct_search_query

tests getting pdf -> getting merge_slide_with_same_headers

assert the structure is correct

```
pdf_doc = extract_words(PDF_NAME)
assert pdf_doc is not None
groupings = text_to_groupings(pdf_doc)
assert groupings is not None
chunks = extract_noun_chunks(groupings)
assert chunks is not None
chunks = merge_slide_with_same_headers(chunks)
assert chunks is not None
chunks = duplicate_word_removal(chunks)
assert chunks is not None

structure_dict = {'Header': str, 'Header_keywords': list, 'Paragraph_keywords': list, 'slides': list}
for slide in chunks:

    for struct in structure_dict.keys():

        assert struct in slide, f'{struct} should be in slide'

        assert slide[struct] is not None, f'{struct} in slide should not be none'

        assert type(slide[struct]) is structure_dict[struct], f'{struct} should be a {structure_dict[struct]}'

def test_construct_search_query():

    pdf_doc = extract_words(PDF_NAME)
    assert pdf_doc is not None
    groupings = text_to_groupings(pdf_doc)
    assert groupings is not None
    chunks = extract_noun_chunks(groupings)
    assert chunks is not None
    chunks = merge_slide_with_same_headers(chunks)
    assert chunks is not None
    chunks = duplicate_word_removal(chunks)
    assert chunks is not None
    query = construct_search_query(chunks)
    assert query is not None

    assert type(query) is list
```