test_integration.py

```python
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
```

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

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

```python
def test_groupings():
```

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

tests getting pdf -> getting groupings

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

check that the structure is right

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

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

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

assert the field is present

```python
            assert struct in slide
```

assert the field is not None

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

assert the type is correct

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

```python
def test_chunks():
```

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

tests getting pdf -> getting groupings

```python
    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)
```

check that the structure is right

```python
    assert chunks is not None
```

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

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

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

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

assert the field is not None

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

assert the type is correct

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

```python
def test_merge_slide_with_same_headers():
```

```
  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

```python
    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)
```

check that the structure is right

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

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

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

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

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

assert the field is present

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

assert the field is not None

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

assert the type is correct

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

```python
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

```python
    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
```

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

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

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

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

assert the field is present

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

assert the field is not None

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

assert the type is correct

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


def test_construct_search_query():
```

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

tests getting pdf -> getting merge_slide_with_same_headers

```python
    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 the structure is correct

```python
    assert type(query) is list
```