Lab 02: Reading Markdown Files

Loading Documents for RAG Systems

CSI403 - Full Stack Program Development

Today's Agenda

Learning:

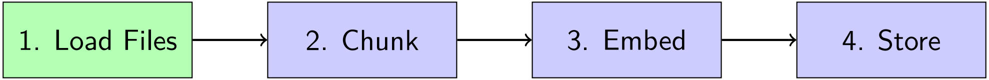
- Why read files?
- Opening files in Python
- Reading content
- Counting and searching
- File encoding (UTF-8)

Time:

- Lecture: 30 min
- Tutorial: 60 min
- Break: 15 min
- Exercise: 45 min
- Submit: 15 min

Why Read Files in RAG?

RAG Pipeline: Step 1 = Load Documents



This lab focuses on Step 1: Loading documents!

- Read Markdown files containing disease information
- Process text content
- Prepare for chunking in Lab 03

Opening Files: The with Statement

Always use `with` statement - it auto-closes the file!

```python
# Good practice: using 'with'
with open("data/rubella.md", "r", encoding="utf-8") as f:
    content = f.read()

# File is automatically closed after the block
print(content)
```

Parameters explained:

- `"data/rubella.md"` - file path
- `"r"` - read mode
- `encoding="utf-8"` - supports Thai characters

Reading Methods

read() - Entire file as string

```
with open("file.md", "r") as f:
    content = f.read()

print(content)
# "Line 1\nLine 2\nLine 3"
```

readlines() - List of lines

```
with open("file.md", "r") as f:
    lines = f.readlines()

print(lines)
# ["Line 1\n", "Line 2\n"]
```

Which to use?

- `read()` - when you need the whole content
- `readlines()` - when processing line by line

Counting Characters and Lines

```python
with open("data/rubella.md", "r", encoding="utf-8") as f:
    content = f.read()

# Count characters
char_count = len(content)
print(f"Characters: {char_count}")

# Count lines
line_count = content.count("\n") + 1
print(f"Lines: {line_count}")

# Or use readlines
with open("data/rubella.md", "r", encoding="utf-8") as f:
    lines = f.readlines()
    print(f"Lines: {len(lines)}")
```

Searching Text in Files

```python
with open("data/rubella.md", "r", encoding="utf-8") as f:
    content = f.read()

# Check if word exists
if "fever" in content:
    print("Found 'fever' in the document!")

# Count occurrences
count = content.count("symptom")
print(f"'symptom' appears {count} times")

# Find lines containing a word
with open("data/rubella.md", "r", encoding="utf-8") as f:
    for line in f:
        if "treatment" in line.lower():
            print(line.strip())
```

File Encoding: UTF-8

UTF-8 encoding is essential for:

- Thai text
- Chinese, Japanese, Korean
- Emoji and special characters

```
# Always specify encoding for non-ASCII text
with open("thai_doc.md", "r", encoding="utf-8") as f:
    content = f.read()

# This will contain Thai text correctly
print(content)  #     "    ..."
```

Important

Without encoding="utf-8", Thai text may appear as garbage characters!

Complete Example: Document Loader

```python
def load_document(filepath):
    """Load a document and return its info."""
    with open(filepath, "r", encoding="utf-8") as f:
        content = f.read()

    return {
        "filepath": filepath,
        "content": content,
        "char_count": len(content),
        "line_count": content.count("\n") + 1
    }

# Use it
doc = load_document("data/rubella.md")
print(f"File: {doc['filepath']}")
print(f"Characters: {doc['char_count']}")
print(f"Lines: {doc['line_count']}")
```

Summary

What we learned:

- `open()` with `with` statement
- `read()` vs `readlines()`
- Counting: `len()`, `count()`
- Searching: `in`, `count()`
- UTF-8 encoding for Thai

Connection to RAG:

- Load documents from files
- Next: Chunk the text
- Later: Create embeddings
- Finally: Store in vector DB

Next Lab

Lab 03: Text Chunking - splitting documents into smaller pieces!

Exercise Preview (4 exercises, 100 points)

1. **Read file content** (25 pts)
   Load rubella.md and store content in a variable
2. **Count characters** (25 pts)
   Count total characters in the file
3. **Count lines** (25 pts)
   Count total lines in the file
4. **Find text** (25 pts)
   Count how many times "symptom" appears

Time: 45 minutes

Work on `exercise/Lab02_Exercise.ipynb`

Questions?

Let's start the Tutorial!

Open: `tutorial/Lab02_Tutorial.ipynb`