Python Programming for Data Science

Unit 15 – Text and document data processing

Dr. Binh Nguyen
Phenikaa School of Computing
binh.nguyenthanh@phenikaa-uni.edu.vn

Learning Objectives

1. Master Python libraries for text processing
2. Learn document parsing techniques
3. Implement practical text analysis projects

Why Text Processing Matters

🔍 Data Analysis
- Extract insights from unstructured text data
- Process social media posts, reviews, and surveys

🤖 Machine Learning
- Prepare text data for NLP models
- Enable sentiment analysis and classification tasks

📄 Document Management
- Automate processing of PDFs and Word documents
- Handle multiple file formats efficiently

🔧 Automation
- Build tools for content extraction
- Automate text cleaning and data transformation

Essential Python Libraries

PHENIKAA UNIVERSITY

Built-in Libraries

- **re** - Regular expressions
- **string** - String operations
- **os** - File system operations

Text Processing

- **pandas** - Data manipulation
- **nltk** - Natural language toolkit
- **spacy** - Advanced NLP

Document Formats

- **PyPDF2/pdfplumber** - PDF processing
- **python-docx** - Word documents
- **openpyxl** - Excel files

Installation:

```
bash
pip install pandas nltk spacy
pip install PyPDF2 pdfplumber python-docx openpyxl
pip install beautifulsoup4 requests
```

Basic Text Operations

PHENIKAA UNIVERSITY

String Manipulation Fundamentals

```
# Basic string operations
text = " Hello World! This is Python Text Processing. "
# Cleaning
clean_text = text.strip().lower()
print(clean_text)
# "hello world! this is python text processing."
# Splitting and joining
words = clean_text.split() print(words)
# ['hello', 'world!', 'this', 'is', 'python', 'text', 'processing.']
# Joining back
rejoined = " ".join(words) print(rejoined)
```

Key Operations:

- strip() - Remove whitespace
- lower()/upper() - Case conversion
- split()/join() - Break apart and combine
- replace() - Text substitution

Regular Expressions (Regex)

PHENIKAA UNIVERSITY

Powerful pattern matching for text processing

```
import re text = "Contact us at john@email.com or call (555) 123-4567"
# Find email addresses
emails = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
print(emails) # ['john@email.com']
# Find phone numbers
phones = re.findall(r'\(\d{3}\)\s\d{3}-\d{4}', text)
print(phones) # ['(555) 123-4567']
# Replace sensitive info
cleaned = re.sub(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', '[EMAIL]', text)
print(cleaned) # "Contact us at [EMAIL] or call (555) 123-4567"
```

Common Patterns:

- \d - Digits
- \w - Word characters
- \s - Whitespace
- + - One or more
- * - Zero or more

Reading Different File Types

PHENIKAA UNIVERSITY

Text Files

```
# Reading plain text files
with open('document.txt', 'r', encoding='utf-8') as file:
        content = file.read()
# Reading line by line
with open('document.txt', 'r', encoding='utf-8') as file:
        for line_num, line in enumerate(file, 1):
                print(f"Line {line_num}: {line.strip()}")
```

JSON Files

python

```
import json
# Read JSON data with open('iris.json', 'r') as file:
        data = json.load(file)
# Extract text from nested structure
texts = [item['content'] for item in data['articles']]
```

CSV Files with Pandas

```
import pandas as pd
# Read CSV with text data
df = pd.read_csv(iris.csv')
print(df.head())
# Access text column
reviews = df['review_text'].tolist()
```

Document Processing

PDF Processing

```
import pdfplumber
# Extract text from PDF
def extract_pdf_text(pdf_path):
        text = ""
        with pdfplumber.open(pdf_path) as pdf:
                for page in pdf.pages:
                        text += page.extract_text() + "\n"
        return text
# Usage
pdf_text = extract_pdf_text(Messi.pdf’)
print(f"Extracted {len(pdf_text)} characters")
```

Document Processing

PHENIKAA UNIVERSITY

Word Documents

```
from docx import Document
# Read Word document
def extract_docx_text(docx_path):
        doc = Document(docx_path) text = []
        for paragraph in doc.paragraphs:
                text.append(paragraph.text)
        return '\n'.join(text)

# Usage
word_text = extract_docx_text(Messi.docx')
```

Text Cleaning & Preprocessing

```
import re
import string
def clean_text(text):
        """Comprehensive text cleaning function"""
        # Convert to lowercase
        text = text.lower()
        # Remove URLs
        text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
        # Remove email addresses
        text = re.sub(r'\S+@\S+', '', text)
        # Remove punctuation
        text = text.translate(str.maketrans('', '', string.punctuation))
        # Remove extra whitespace
        text = re.sub(r'\s+', ' ', text).strip()
        # Remove numbers (optional)
        text = re.sub(r'\d+', '', text)
        return text
# Example usage
dirty_text = "Check out https://example.com! Email: test@email.com. Price: $99.99!!!"
clean = clean_text(dirty_text)
print(clean) # "check out email price"
```

Common Cleaning Steps:

- Remove URLs and email addresses
- Handle punctuation and special characters
- Normalize whitespace
- Convert to consistent case

Text Analysis with NLTK

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from collections import Counter

# Download required data (run once)
# nltk.download('punkt')
# nltk.download('stopwords')
def analyze_text(text):
# Tokenization
words = word_tokenize(text.lower())
sentences = sent_tokenize(text)
# Remove stopwords stop_words =
set(stopwords.words('english'))
filtered_words = [word for word in words if word not in
stop_words]
# Stemming
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word)
for word in filtered_words]
        # Word frequency
        word_freq = Counter(filtered_words)
        return {
                'word_count': len(words),
                'sentence_count': len(sentences),
                'unique_words': len(set(words)),
                'top_words': word_freq.most_common(5)
        }
# Example
text = "Python is amazing for text processing. Text processing
with Python is powerful."
analysis = analyze_text(text)
print(analysis)
```

Advanced Text Processing

PHENIKAA UNIVERSITY

Text Similarity

```
from sklearn.feature_extraction.text import TfidfVectorizer from
sklearn.metrics.pairwise import cosine_similarity

def text_similarity(text1, text2):
        """Calculate similarity between two texts"""
        vectorizer = TfidfVectorizer()
        tfidf_matrix = vectorizer.fit_transform([text1, text2])
        similarity = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
        return similarity[0][0]

# Example
doc1 = "Python is great for data science"
doc2 = "Data science with Python is awesome"
similarity = text_similarity(doc1, doc2)
print(f"Similarity: {similarity:.3f}")
```

Advanced Text Processing

Named Entity Recognition with spaCy

```
import spacy
# Load English model (install with: python -m spacy download en_core_web_sm)
nlp = spacy.load("en_core_web_sm")
def extract_entities(text):
        doc = nlp(text)
        entities = []
        for ent in doc.ents:
                entities.append({
                        'text': ent.text,
                        'label': ent.label_,
                        'description': spacy.explain(ent.label_)
                })
        return entities

# Example
text = "Apple Inc. was founded by Steve Jobs in Cupertino, California."
entities = extract_entities(text)
for entity in entities:
        print(f"{entity['text']} -> {entity['label']} ({entity['description']})")
```

Best Practices & Next Steps

🎯 Best Practices

- Always handle encoding (UTF-8)
- Use context managers for file handling
- Validate input data
- Handle errors gracefully
- Document your text processing pipeline

⚡ Performance Tips

- Process large files in chunks
- Use generators for memory efficiency
- Cache compiled regex patterns
- Consider parallel processing
- Profile your code

🚀 Next Steps

- Explore machine learning with text
- Learn about transformers (BERT, GPT)
- Build text classification models
- Create chatbots and QA systems
- Practice with real datasets

Key Takeaways

1. Python offers powerful tools for text processing
2. Start with basic operations, then advance to NLP
3. Always clean and preprocess your text data
4. Practice with real-world projects and datasets

THANK YOU!