Pipeline Design, Strings, Evaluation

Problem Solving using Python - Week 5

Homework and Last Week Q&A

Learning Objectives

Learning Objectives

At the end of this lecture, you will...

Learning Objectives

At the end of this lecture, you will...

1. solve programming problems using a *pipeline* design.

Learning Objectives

At the end of this lecture, you will...

1. solve programming problems using a *pipeline* design.
2. perform string manipulations on a structured file using string methods (`split`, `join`, `format`).

Learning Objectives

At the end of this lecture, you will...

1. solve programming problems using a *pipeline* design.
2. perform string manipulations on a structured file using string methods (`split`, `join`, `format`).
3. evaluate the *design and code* aspects of your program.

Three Problems

Three Problems

Text Pre-Processing

- **Input:** collection of texts (`list` of `str`)
- **Output:** collection of tokens (`list` of `list` of `str`)
- **Steps:** remove empty strings, remove duplicates, tokenize, lower-case, vocabulary restriction

Three Problems

Text Pre-Processing

- **Input:** collection of texts (`list` of `str`)
- **Output:** collection of tokens (`list` of `list` of `str`)
- **Steps:** remove empty strings, remove duplicates, tokenize, lower-case, vocabulary restriction

Data Pre-Processing

- **Input:** two tables of numeric data (two `list` of `list` of `float`)
- **Output:** one table (`list` of `list` of `float`)
- **Steps:** remove duplicates, merge two tables by shared column, group by column, calculate mean per group

Three Problems

Text Pre-Processing

- **Input:** collection of texts (`list` of `str`)
- **Output:** collection of tokens (`list` of `list` of `str`)
- **Steps:** remove empty strings, remove duplicates, tokenize, lower-case, vocabulary restriction

Data Pre-Processing

- **Input:** two tables of numeric data (two `list` of `list` of `float`)
- **Output:** one table (`list` of `list` of `float`)
- **Steps:** remove duplicates, merge two tables by shared column, group by column, calculate mean per group

Image Pre-Processing

- **Input:** image (=2D pixels, `list` of `list` of `list` of `int`)
- **Output:** standardized image
- **Steps:** resize/crop to a fix size, balance brightness, grayscale conversion

What do all these problems have in common?

What do all these problems have in common?

What do all these problems have in common?

The process consists of a series of steps

i.e., the output of the previous step

is the input of the next step

What do all these problems have in common?

The process consists of a series of steps

i.e., the output of the previous step

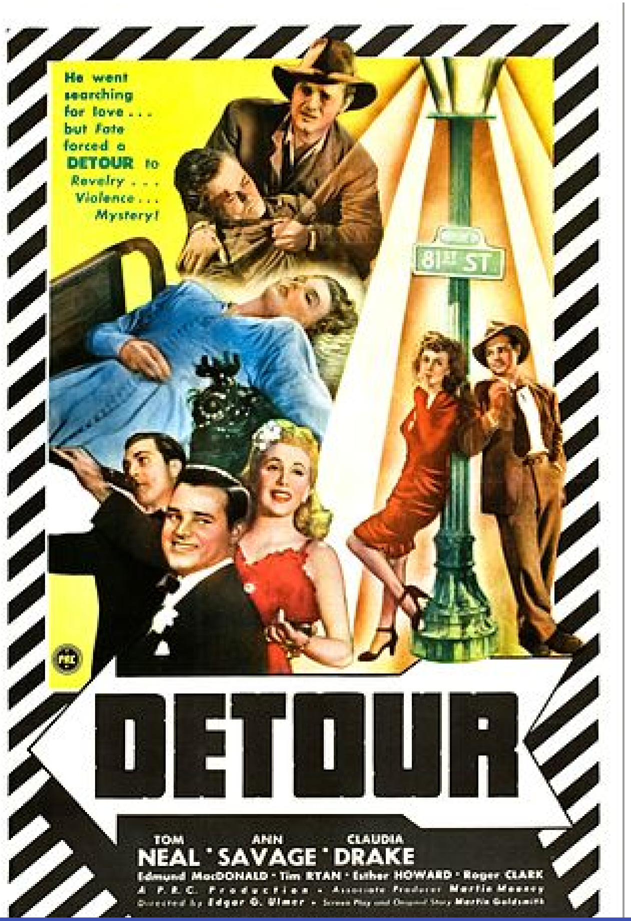
is the input of the next step

Pipeline Design

Subtitles Synchronization

Problem: Subtitles Synchronization

Problem: Subtitles Synchronization


He went searching for love... but Fate forced a DETOUR to Revelry... Violence... Mystery!
81ST ST
DETOUR
TOM NEAL · ANN SAVAGE · CLAUDIA DRAKE
Edmund MacDONALD · Tim RYAN · Esther HOWARD · Roger CLARK
A P.R.C. Production · Associate Producer Martin Mooney
Directed by Edgar G. Ulmer · Screen Play and Original Story Martin Goldsmith

Detour is a 1945 American film noir directed by Edgar G. Ulmer starring Tom Neal and Ann Savage.

In 1992, Detour was selected for preservation in the United States National Film Registry by the Library of Congress as being "culturally, historically, or aesthetically significant".

The film is in the **public domain** and is freely available from online sources.

Source: Wikipedia

Problem: Subtitles Synchronization

Goal

Sync the subtitles to the video

i.e., shift in time the appearance of the subtitles

(e.g., 2 seconds forward)

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Incremental Development

1. Reinterpret the Problem

1. Reinterpret the Problem

Input

Original `str` file format

1. Reinterpret the Problem

Input

Original `str` file format

Output

Shifted `str` file format

1. Reinterpret the Problem

Input

Original `str` file format

Output

Shifted `str` file format

What's this `str` file format?

`str` file format

`str` file format

```
...
29
00:02:56,460 --> 00:02:58,330
Hey, turn that off.
Will you turn that thing off?

30
00:02:58,380 --> 00:03:00,210
- What’s eating you now?
- Yeah, what’s eating you?

31
00:03:00,250 --> 00:03:02,300
- That music, it stinks.
- Oh, you don’t like it, huh?
...
```

`str` file format

```
...
29
00:02:56,460 --> 00:02:58,330
Hey, turn that off.
Will you turn that thing off?

30
00:02:58,380 --> 00:03:00,210
- What’s eating you now?
- Yeah, what’s eating you?

31
00:03:00,250 --> 00:03:02,300
- That music, it stinks.
- Oh, you don’t like it, huh?
...
```

Every subtitle quote made up of few lines:

1. First line
   `index`
2. Second line - timing
   `<start_time> --> <end_time>`
3. Third line (and sometimes fourth) - the text itself

`str` file format

```
...
29
00:02:56,460 --> 00:02:58,330
Hey, turn that off.
Will you turn that thing off?

30
00:02:58,380 --> 00:03:00,210
- What’s eating you now?
- Yeah, what’s eating you?

31
00:03:00,250 --> 00:03:02,300
- That music, it stinks.
- Oh, you don’t like it, huh?
...
```

Every subtitle quote made up of few lines:

1. ➡ First line
   `index`
2. Second line - timing
   `<start_time> --> <end_time>`
3. Third line (and sometimes fourth) - the text itself

`str` file format

```
...
29
00:02:56,460 --> 00:02:58,330
Hey, turn that off.
Will you turn that thing off?

30
00:02:58,380 --> 00:03:00,210
- What’s eating you now?
- Yeah, what’s eating you?

31
00:03:00,250 --> 00:03:02,300
- That music, it stinks.
- Oh, you don’t like it, huh?
...
```

Every subtitle quote made up of few lines:

1. First line
   `index`
2. ➡ Second line - timing
   `<start_time> --> <end_time>`
3. Third line (and sometimes fourth) - the text itself

`str` file format

```
...
29
00:02:56,460 --> 00:02:58,330
Hey, turn that off.
Will you turn that thing off?

30
00:02:58,380 --> 00:03:00,210
- What’s eating you now?
- Yeah, what’s eating you?

31
00:03:00,250 --> 00:03:02,300
- That music, it stinks.
- Oh, you don’t like it, huh?
...
```

Every subtitle quote made up of few lines:

1. First line
   `index`
2. Second line - timing
   `<start_time> --> <end_time>`
3. ➡ Third line (and sometimes fourth) - the text itself

2. Design a Solution

2. Design a Solution

Does this problem require a pipeline design?

2. Design a Solution - Design Strategy

2. Design a Solution - Design Strategy

How to come up with a design,

i.e. breaking the problem to sub-problems?

2. Design a Solution - Design Strategy

How to come up with a design,

i.e. breaking the problem to sub-problems?

Top-Down Strategy

1. *Split* `.str` file content into quotes
2. *Split* each subtitle quote to its lines
3. *Take* the second line (`timing`)
4. *Split* to start and and end timing
5. *Add* the `shift` to each of the timing
6. *Join* the two timing into one line
7. *Join* the lines into quotes

2. Design a Solution - Design Strategy

How to come up with a design,

i.e. breaking the problem to sub-problems?

Top-Down Strategy

1. *Split* `.str` file content into quotes
2. *Split* each subtitle quote to its lines
3. *Take* the second line (`timing`)
4. *Split* to start and and end timing
5. *Add* the `shift` to each of the timing
6. *Join* the two timing into one line
7. *Join* the lines into quotes

We will solve first the "smaller"/"internal" sub-problems and the "bigger"/"external" ones,

because this makes it easier to solve **this problem incrementally**

Jupyter Notebook!

From Jupyter Notebook
To a Python Script

Pipeline Design
Wrap-up + Q&A

(this is not the end yet)

Evaluate Phase

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Evaluate Phase

Outcome - Code

Evaluate Phase

Outcome - Code

Evaluation Criteria

1. Functionality
2. ➡ Design and code
3. Readability, style & documentation

Evaluate Phase - Design and Code

1. Structure and flow
2. Modularization
3. Data structure
4. Idiomatic python

Evaluate Phase - Design and Code - Structure and Flow

```
def is_sorted_v1(seq):
    """Check whether a sequence is ordered or not."""

    result = True

    for i in range(len(seq)):
        for j in range(i+1, len(seq)):
            is_pair_ordered = (seq[i] <= seq[j])
            result = result and is_pair_ordered

    return result
```

Evaluate Phase - Design and Code - Structure and Flow

```python
def is_sorted_v2(seq):
    """Check whether a sequence is ordered or not."""

    result = True
    i = 0
    for i in range(len(seq)-1):
        is_pair_ordered = (seq[i] <= seq[i+1])
        result = result and is_pair_ordered

    return result
```

Evaluate Phase - Design and Code - Modularization

```python
def calc_mean_difference_v1(first_group, second_group):
    """Calculate the mean difference between two groups."""

    # calculate the mean of the first group
    first_total = 0
    for item in first_group:
        first_total += item
    first_mean = first_total / len(first_group)

    # calculate the mean of the second group
    second_total = 0
    for item in second_group:
        second_total += item
    second_mean = second_total / len(second_group)

    return first_mean - second_mean
```

Evaluate Phase - Design and Code - Modularization

```python
def calc_mean(group):
    "Calculate the mean of a group."
    total = 0
    for item in group:
        total += item
    return item / len(group)

def calc_mean_difference_v2(first_group, second_group):
    """Calculate the mean difference between two groups."""
    first_mean = calc_mean(first_group)
    second_mean = calc_mean(second_group)
    return first_mean - second_mean
```

Evaluate Phase - Design and Code - Data Structure

```
country2capital = [('Germany', 'Berlin'),
                   ('Japan', 'Tokyo'),
                   ('Cuba', 'Havana')]

for item in country2capital:
    if item[0] == 'Cuba':
        print(item[1])
```

Evaluate Phase - Design and Code - Data Structure

```
country2capital = {'Germany': 'Berlin',
                   'Japan': 'Tokyo',
                   'Cuba': 'Havana'}

print(country2capital['Cuba'])
```

Evaluate Phase - Design and Code - Idiomatic Python - 1

```python
def join_with_comma(strings):
    """Join a list of strings into one string with comma."""
    line = ''

    for s in strings[:-1]:
        line += s + ','

    if strings:
        line += strings[-1]

    return line
```

Evaluate Phase - Design and Code - Idiomatic Python - 1

```
','.join(['one', 'two', 'three'])
```

Evaluate Phase - Design and Code - Idiomatic Python - 2

```
def sum_v1(seq):
    """Sum the elements of a list of numbers."""
    total = 0
    for i in range(len(seq)):
        total += seq[i]
    return i
```

Evaluate Phase - Design and Code - Idiomatic Python - 2

```
def sum_v2(seq):
    """Sum the elements of a list of numbers."""
    total = 0
    for item in seq:
        total += item
    return i
```

Evaluate Phase - Design and Code - Idiomatic Python - 2

```
sum([1, 3, 5])
```

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Wrap-up + Q&A

Problem Solving using Python - Week 5

Pipeline Design, Strings, Evaluation