



Data Science APIs are complex

Data Science APIs like Pandas, SkLearn have a huge number of functions and arguments

We are looking into using program summarization to aid developers in understanding such programs

```
DataFrame.replace(to_replace=None, value=_NoDefault.no_default, *,  
inplace=False, limit=None, regex=False, method=_NoDefault.no_default)  
Replace values given in to_replace with value.  
Values of the DataFrame are replaced with other values dynamically.  
This differs from updating with .loc or .iloc, which require you to specify a location to  
update with some value.  
Parameters: to_replace : str, regex, list, dict, Series, int, float, or None  
How to find the values that will be replaced.  
• numeric, str or regex:  
    ○ numeric: numeric values equal to to_replace will be replaced  
        with value  
    ○ str: string exactly matching to_replace will be replaced with  
        value  
    ○ regex: regex matching to_replace will be replaced with value  
• list of str, regex, or numeric:  
    ○ First, if to_replace and value are both lists, they must be the  
        same length.  
    ○ Second, if regex=True, then all the strings in both lists will  
        be interpreted as regexes otherwise they will match directly. This  
        doesn't matter much for value since there are only a few  
        possible substitution regexes you can use.  
    ○ str, regex and numeric rules apply as above.
```

## Motivation

### Issues in Language Models for Program Summarization

Robustness to perturbations in programs

Summarize the following pandas program imperatively in a line  
`'df.drop_duplicates(subset=['roll_number'], keep='first')`

This Pandas program drops duplicate rows from a DataFrame based on the 'roll\_number' column and keeps only the first occurrence of each unique 'roll\_number'.

Summarize the following pandas program imperatively in a line  
`'df.drop_duplicates(subset=1, keep='first')`

This program drops all the duplicate rows in DataFrame 'df' based on the values in the second column while keeping the first occurrence of each unique value.

Compositional Generation

Summarize the following pandas program imperatively in a line  
`'df.replace('a':0, 'b':1, 'c':2, 'd':3, 'a':100, 'b':200, 'c':300, 'd':400)'`

The program replaces the values 'a' with 0, 'b' with 1, 'c' with 2, and 'd' with 3 in the pandas DataFrame 'df', and then replaces the same values with 'a' with 100, 'b' with 200, 'c' with 300, and 'd' with 400, respectively.

Controllability

These systems provide very little control to end users as to how templates should look

They can generate incorrect answers

## Parametric Templates – Using API knowledge for summarization

### Defining Parametric Templates

Given an API function  $n$  along with signature  $t_1, t_2, \dots, t_n$

Parametric Templates are defined by a sequence of

- words ( $w_i$ )
- holes function (mapping from arguments to sequence of words)

### Learning Parametric Templates

Writing parametric templates is not easy!

requires domain expertise

templates are fuzzy and hard to manually annotate

numerous functions and arguments!

### Parametric Templates Examples

`df[df['score'].isin(range(5,10))]`

Select the rows where value in column score lie in the integers between 5 and 10 (exclusive)

- Template[subscript, [caller - df, arg - expr]]
  - Select the rows VAR1
  - VAR1 = Summary(`df['score'].isin(range(5,10))`)
- Template[isin, [caller - df['score'], value - range(5,10)]]
  - where values in column score lie in VAR2, where
  - VAR1 = Summary(`range(5,10)`)
- Template[range, [start - int, end - int]]
  - the integers between start and end(exclusive)

### Learned Templates

`df.replace({'country': {'Germany': 'GER', 'France': 'FRA'}})`

Replace the values  $F^1$  in column  $F^2$  with  $F^3$  respectively

$F^1(P) = \text{'Germany'}$  and  $\text{'France'}$

$F^2(P) = \text{'country'}$

$F^3(P) = \text{'GER'}$  and  $\text{'FRA'}$

`df.replace({'a':1, 'b':2, 'c':3}, {'a':100, 'b':200, 'c':300})`

Replace the values  $F^1$  with  $F^3$  respectively

$F^1(P) = 1$  in  $\text{'a'}$ , 2 in  $\text{'b'}$ , 3 in  $\text{'c'}$

$F^2(P) = 100, 200, 300$

`df.dropna(subset=['score1', 'score2', 'score3'], thresh=2)`

Drop the rows in df having atleast  $F^1$  nans in the  $F^2$  columns

$F^1(P) = 2$

$F^2(P) = \text{'score1', 'score2', and 'score3'}$

We learn templates from corpus of API snippets and summaries

- dynamic programming on spans of texts
- bottom-up program synthesis
- word and phrase-based similarity