

CS23M117

V.NIKHITHA

M.TECH CSE

SOFTWARE DOCUMENTATION USING CODE SUMMARIZATION (SDCS)

PROBLEM STATEMENT:

If the software has no proper documentation, it is very difficult to understand large software. But preparing manually consumes more time.

MY SOLUTION TO THE PROBLEM:

Automate software documentation with deep learning models. SDCS is a software documentation tool for python projects.

LIBRARIES USED:

- Google-colab is used to develop this tool.
- GPU Runtime type is necessary to run finetuning source code.
- Datasets and Transformer libraries are needed to be installed.

This tool documents a project by summarising individual methods by a finetuned text summarization model and all the generated summaries are input to a BART transformer to generate source code documentation and summaries of all source codes are used to generate project level documentation.

FINETUNING:

- It imports necessary libraries and loads the CodeXGLUE dataset for Python, selecting a subset for training facebook/bart-base summariser.

- The BART tokenizer and model for conditional generation are initialized from the "facebook/bart-base" pre-trained checkpoint.
- Optimizer (Adam) and loss function (CrossEntropyLoss) are used for training.
- A custom dataset class is defined to process input code and target summaries, tokenizing them appropriately.
- A DataLoader is created for efficient training with batch size, shuffling, and parallel data loading settings.
- The fine-tuned model is saved to the specified directory after training completion.

This finetuned model is test on CodeXGLUE and CODE_SEARCH_NET datasets, rouge score is similar in both cases.

- ROUGE-1: This metric evaluates the overlap of unigrams (individual words) between the generated and reference summaries. The reported scores indicate:
Recall (r): 65.87%
Precision (p): 98.24%
F1-score (f): 72.99%
- ROUGE-2: This metric evaluates the overlap of bigrams (pairs of consecutive words) between the generated and reference summaries. The reported scores are:
Recall (r): 62.42%
Precision (p): 97.66%
F1-score (f): 69.07%
- ROUGE-L: This metric considers the longest common subsequence (LCS) between the generated and reference summaries, which emphasizes sentence-level structure. The reported scores are:

Recall (r): 65.87%

Precision (p): 98.24%

F1-score (f): 72.99%

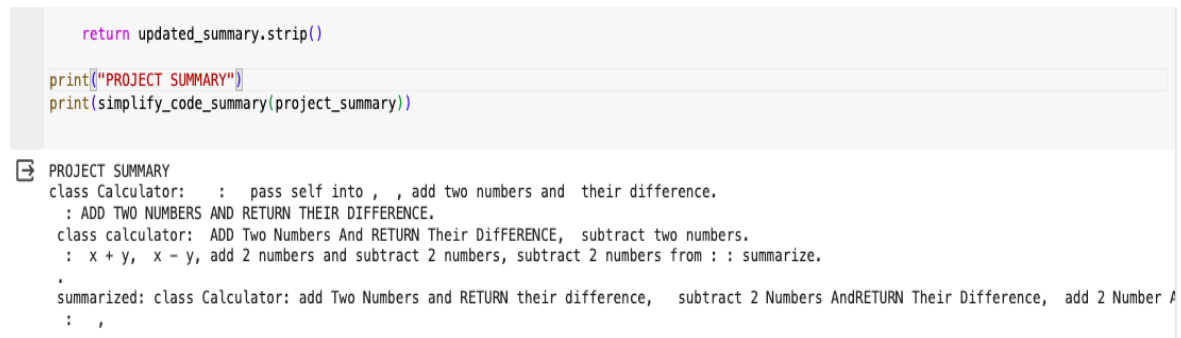
- Since methods are summarized with high accuracy it guarantees the accuracy of the software documentation.

RESULTS:

SDCS is used to document a small project calculator (EXAMPLE_PYTHON_PROJECTS folder). And the generated summary is

```
return updated_summary.strip()

print("PROJECT SUMMARY")
print(simplify_code_summary(project_summary))
```



```
PROJECT SUMMARY
class Calculator: : pass self into , , add two numbers and their difference.
: ADD TWO NUMBERS AND RETURN THEIR DIFFERENCE.
class calculator: ADD Two Numbers And RETURN Their Difference, subtract two numbers.
: x + y, x - y, add 2 numbers and subtract 2 numbers, subtract 2 numbers from : : summarize.
.
summarized: class Calculator: add Two Numbers and RETURN their difference, subtract 2 Numbers AndRETURN Their Difference, add 2 Number A
: ,
```

SDCS is used to document a project ChessGame (EXAMPLE_PYTHON_PROJECTS folder). And the generated summary is

```
print("PROJECT SUMMARY")
print(simplify_code_summary(project_summary))
```

Enter the path of the fine-tuned model: /content/drive/MyDrive/finetuned_model
Enter the path of the folder containing Python source files: /content/drive/MyDrive/chess_game
PROJECT SUMMARY
The game engine scales chess pieces to fit the square size of the board.
The game state is displayed as a black and white image of the chess board.
It is used to show the state of the game as well as the current state of each piece.
It also shows the current position of the player and their position on the board at the start and end of a game.
It can also be used to display the position of different pieces at different times in the game, such as when a move is made or a new move is made.
For more information on how to play chess in the UK, visit <https://www.bbc.com/sport/games-and-hobbies/2019/08/20190815-chess-uk> or go to <https://www.bbc.com/sport/games-and-hobbies/2019/08/20190815-chess-uk>.
In the US and Canada, go to <https://www.uschess.org/> or call the National Chess Association on 1-800-273-8255 or visit a local branch of the NCA.
In Europe, the NCCA is based at the University of Edinburgh.

HOW TO IMPLEMENT THE PROJECT:

- Open google colab and clone the repository
- Finetuning model is of 2GB capacity not able to upload in github so this is the google drive link for finetuned_model

https://drive.google.com/drive/folders/1aBWZkUKxzBsDpVMc47NUUQOzjB_mGltj?usp=drive_link

- It can be downloaded from the link, as a zip file and can be added to own drive.
- Otherwise finetuned_final.ipynb should be run to get finetuned_model in the drive using GPU but takes a long time.
- Change path to SDCS_ipynb_notebooks directory and run SDCS.ipynb
- Input the absolute paths of finetuned model directory and input_folder to be documented during the runtime (without double quotes).

NEW THINGS I LEARNED FROM THE PROJECT:

- Learned about finetuning, BART Transformers and Different datasets.
- Made me familiar to Hugging face and code documentation tools.
- Learned about different libraries and rouge score.
- Tokenizing the data efficiently to avoid completely occupying RAM by the large deep learning model.