EE / CprE / SE 492 Weekly Report 03: SDMay21-29
Intelligent Code Editor

Feb. 23 - Mar. 1
Client: Hung Phan
Advisor: Dr. Ali Jannesari

Team Members & Roles:

Evan Christensen - Meeting Scribe
Ben Gonner - Report Manager
Jacob Puetz - Chief Engineer Software Systems
Jordan Silvers - Meeting Facilitator
Cory Smith - Test Engineer

Weekly Summary:

In the previous week, we continued work on the GNN, plugin, and dataset. The GNN now produces model files that can be loaded later and used without needing to retrain the GNN. Additionally, the output of the GNN has been translated from an array of numbers to a meaningful category string.

Also in the previous week, we completed crucial tutorials on extension development in Visual Studio. Along with the basic “Hello World,” which teaches us the beginnings of the extension development, we also did the Context Menu tutorial. The Context Menu tutorial allows us to add specific commands to the context/right click menu which will be crucial for our plugin. Lastly, we have continued our work on categorizing the dataset. Various loops and variable declarations have been added as categories.

Past Week Accomplishments:

- Visual Studio Plugin Development - Ben/Jordan
  - Researched several tutorials on Visual Studio extensions:
    - Hello World - adds “Hello World” to the tools menu (Figure 1)
    - Context Menu Tutorial - Adds a specific command to the context menu (Figure 2)
  - Research on other areas of task list for extension development:
    - Pass Line or Highlighted text into event handler
    - Add Command to Text Editor Context menu (right click menu)
    - Change the text in the editor from command handler
- GNN Model Exporting & Reuse - Cory
  - The GNN now exports files that can be loaded to use the trained model again
    - It actually produces 4 files: a data, a meta, an index, and a checkpoint file. I think the data file is the main one, but the others (except probably the checkpoint one) may be necessary for loading the model
  - A meaningful classification string can now be made from the GNNs output

        - The GNNs predict method returns an array of numbers between 0 and 1, each indicating the probability of a given classification; The highest probability is used as the classification
        - The index of each probability corresponds to a string in the data/corpus/<dataset>_labels.txt file, which can be used to identify what classification the result is
- Training Data Pre-Processing - Jacob/Evan
    - Loops/Variable declarations have been sorted
        - Folders are dynamically allocated for both pseudocode and code classifications as well
        - Setback slightly as current training data isn't quite satisfactory (see next bullet)
    - Client wants a change in some of the training data's form
        - Need to edit around 1000 lines in the dataset
        - 200 lines per group member have been assigned to make the process less tedious

Individual Contributions:

Name	Contributions	Hours This Week	Total Hours
Evan Christensen	Training Data Pre-Processing	3	11
Ben Gonner	Visual Studio Plugin Development	3	9
Jacob Puetz	Training Data Pre-Processing	2	10
Jordan Silvers	Visual Studio Plugin Development	4	12
Cory Smith	GNN Model Exporting & Reuse	6	20

Upcoming Plans:

- In regards to the extension, we plan on during more research on the areas of passing a line/highlighted text through the event handler
    - We have some information on how to do this, it's just a matter on if the information we attained is up to date/still works
    - We are also working towards being able to change code inside of visual studio based on our extension
- For the reusable model, we hope to be able to encode a string and use it as input for the GNN
    - Currently, the encoding process only works on files with a specific structure, not individual strings, which makes things hard when trying to use it for single predictions

Figure 1:

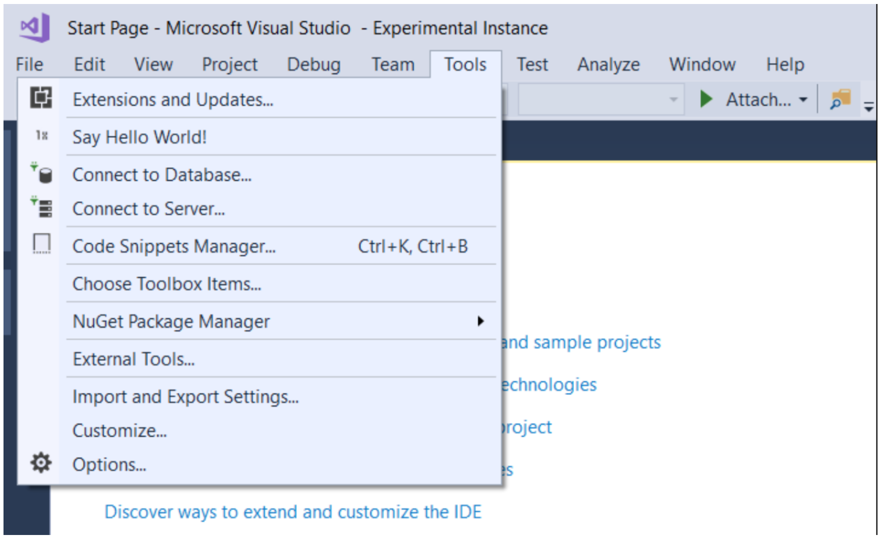
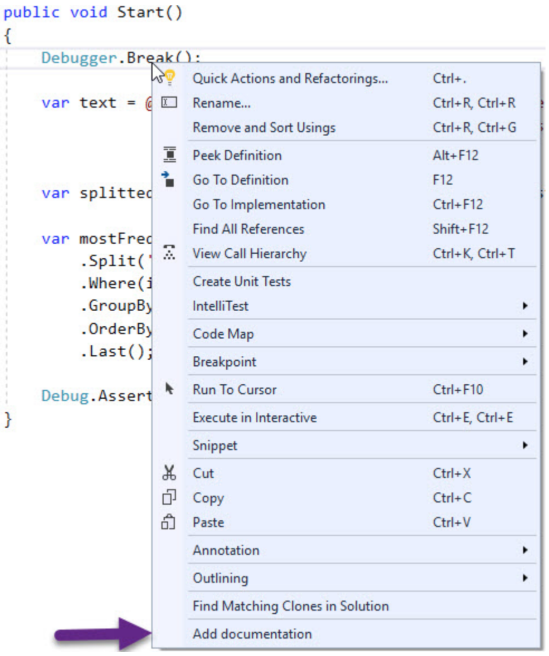


Figure 2:



```
public void Start()
{
    Debugger.Break();

    var text = @


    var splitted

    var mostFreq
        .Split('
        .Where(i
        .GroupBy
        .OrderBy
        .Last();

    Debug.Assert
}
```

Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Implementation Ctrl+F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T
Create Unit Tests
IntelliTest
Code Map
Breakpoint
Run To Cursor Ctrl+F10
Execute in Interactive Ctrl+E, Ctrl+E
Snippet
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Annotation
Outlining
Find Matching Clones in Solution
Add documentation