Python Word Ladder Debugging and Version Control Assignment

2810ICT— Software Technologies
School of ICT
Griffith University

Kane Clerke
S5057278

Josh Mac Guinness
S5057696

Table of Contents

1.0 Problem Statement

The assignment for 2810ICT Software Technologies, looks at the implementation of a word ladder program using regular expressions and basic machine learning. The original program takes an English dictionary text file, that is then used to take a starting word that then can only be altered one character at a time to reach the target word. For the sake of the assignment, the original program takes the original word and target word, 'lead' and 'gold' and converts it in 481 words. Ideally the new program will convert the above to words in 3 steps (lead, load, goad, gold). This will be implemented by converting the original program and optimising its algorithm. The finished program will also additionally find a six-word path from words 'hide' to 'seek'. A user will also be able to enter a series of words that they wish to exclude from the given path, as well as asking for only the shortest path to be given for said word pair.

2.0 User Requirements

The following outlines the user requirements for the program:

- **The user will be able to load in a dictionary of there choosing.**
- **The user will type all input for the program**
- **The user will be able to process the path of any word to any other word as long as said words are found the the supplied dictionary**
- **The user will not be able to enter special characters or numbers for the target words or dictionary file**
- **The user will be able to enter a list of words to not be used in the path of the program**
- **The user may choose to have the shortest path or just the desired path to the word**
- **The user can only enter two words of equal length**
- **The user will be prompted if they have entered any combination of incorrect inputs**

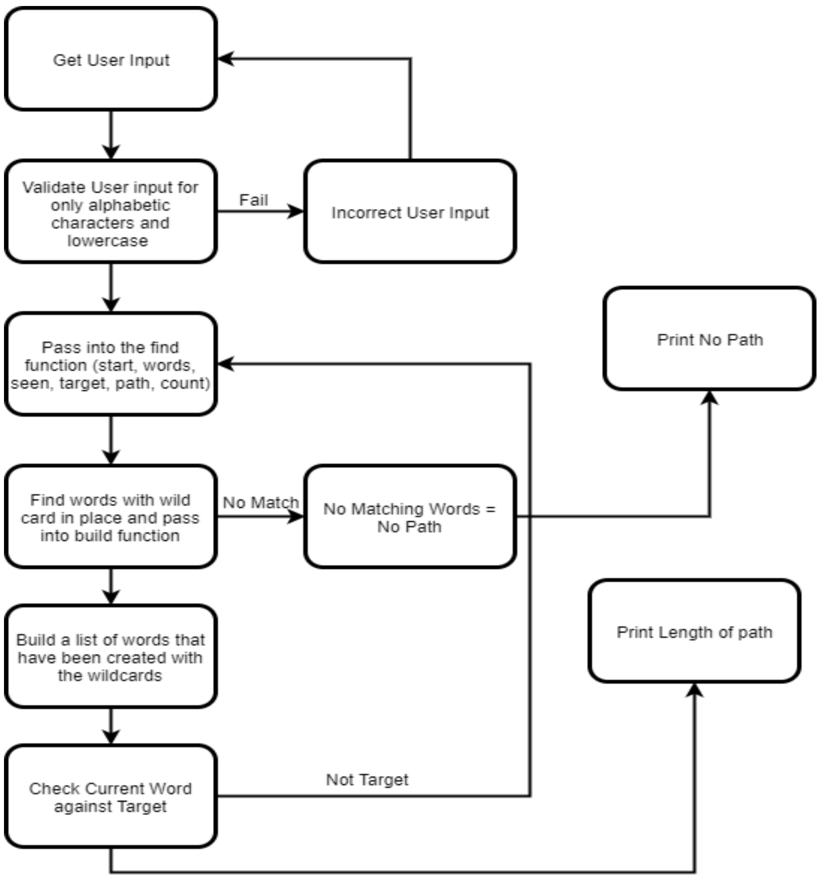
3.0 Software Requirements

The following outlines the software requirements for the program:

- **The program will read a dictionary text file supplied by the user**
- **The program will take keyboard only input from user**
- **The program will only make changes to the word one letter at a time**
- **The program will only allow words that match a list supplied by user called dictionary.txt**
- **The program shall only use a word in the path as long as it has not been exclude from the list after user input.**
- **The program will allow for text input**
- **The program will go from words lead to gold in 3 steps and hide to seek in 6.**
- **The program will be unit test compliant.**
- **The program will be version controlled using GitHub**
- **The program will only take two words of same length to find a path**
- **The program will only take alphabetical character (no special characters or numbers)**

4.0 Software Design

Logical Block Diagram


Get User Input
Validate User input for only alphabetic characters and lowercase
Fail
Incorrect User Input
Pass into the find function (start, words, seen, target, path, count)
Print No Path
Find words with wild card in place and pass into build function
No Match
No Matching Words = No Path
Build a list of words that have been created with the wildcards
Print Length of path
Check Current Word against Target
Not Target

List of all functions in the software:

Function: same(item, target):

- This function is the simplist of the main three function and simply returns how many charecters match between the item and the target word. For example lead and gold would return 1 while goad and gold would return 3

Input Parameters

- Item (String) – The word at the closest rung on the ladder.
- Target (String) – The target word as dictated by the user.

Side Effects

- None

Return Value

- Integer - Returns the integer value to that of the number of common letters between the start and end words.

Function: find(word, words, seen, target, path, count=1):

- The find function finds the path from the starting word to the target word by building a list of words using a "." as a wild card in order to test different combinations of rearangements. It then steps thourgh how many charecters match the target word using tuple's that are then returned to find the most suitable word. Finally if the start and target match the loop will terminate

Input Parameters

- Seen (Dictionay) – Boolean values, words that have already been processed by the program.
- Target (String) – The target word as dictated by the user.
- Words (List) – A list of words generated from the dictionary file.

Side Effects

- None

Return Value

- String – If both starting and target words matched the loop, program is ended.

Function: build(pattern, words, seen, list):

- Build takes in the pattern, the list of seen words and the test words and then complies a list of words using the wildcard "." system to then be searched using regular expression search which is a pre defined function with the re library.

Input Parameters

- Pattern (String) – A string that is matched by the function alongside 'words' using a "." which is the wildcard system.
- Words (List) – A list of words generated from the dictionary file.
- Seen (Dictionay) – Boolean values, words that have already been processed by the program.
- List (List) – Words that are to be used for the next rung in the word ladder that have been identified.

Side Effects

- None

Return Value

- List – Returns a list of every possible word within the given dictionary that match the pattern.

Pseudocode for non-standard algorithms

```
Find(word, words, seen, target, path, count=1):
    For (I in the length of the word)
        Build a list of words using '.' As a wild card
    Check to see if there is a path
    Find how many letters match the target in the new word
    If  (length of created word matching characters = target word)
        Leave the loop as the words match

Build(pattern, words, seen, list):
    For (the words in the word list)
        If (the regular expression search & the word hasn't already been used)
            Return the word

Same(item, Target):
    Return the length of the two combined target and current words
    Check to see if the current word is equal to the final word
    Words match then leave function
```

List of all data structures in the software

Dictionaries

- `seen = {start : True}`
    - Used in the seen function and used to call and check if the current word and target word match to print the final path length output

List

- `[c for (c, t) in zip(item, target) if c == t]`
    - Used for the storing of the len output of a given variable and then passed to other functions
    - Used in function same
    - Item is the current word, Target is the final goal word, C/T are used to step through the array
- `list = []`
    - Creates a blank list that is then later used and has the list of wild card words passed into it for checking later in the program
    - Used in function find and build
    - List is also over written multiple times passing in a vast range of different variables
- `words = []`
    - Appends the original list of words and dictionary to a list
    - Used in function Find

Tuples

- `list = sorted([(same(w, target), w) for w in list])`
    - Combination of both a list and a tuple this line of code returns all the tuples of how many matched characters there are from the current word to final word
    - Used in all functions

Strings

- `start = input("Enter start word:").lower()`
    - Used to take user input for the initial word. Similar code also takes input for the dictionary and the target word.
- `print("No path found")`
    - Used to print the path of the word if none is found. Simmilar to the same code for printing the path if one is found. `print(len(path) - 1, path)`

Configuration management and version control

GitHub was our groups preferred method of version control and allowed us to integrate with PyCharm, an easy to use python interpreter. By using GitHub we were able to both work on files at the same time and have a collaborative work flow while having the ability to back track to any old code if something was to happen. It also gave us the safety of cloud hosting. Find the GitHub at the link below and look through the commits to the repository to see the evolution of our code.
Josh Mac Guinness – Joshjpm
Kane Clerke – valkyriak.

Find a complete log of all GitHub integrations at the end of the document
Commits on Aug 20, 2017

5.0 Unit Tests

Same Function Tests			
Number	**Test Case**	**Expected Results**	**Actual Results**
1.1	Test 1 matching char	True: Pass Test	True Pass Test
1.2	Test multiple matching char	True: Pass Test	True Pass Test
1.3	Test same word match	True: Pass Test	True Pass Test
1.4	Test 0 match words	True: Pass Test	True Pass Test

Find Functions Test			
Number	**Test Case**	**Expected Results**	**Actual Results**
2.1	Test found path	True: Pass Test	True: Pass Test
2.2	Test Is none case	True: Pass Test	True: Pass Test
2.3	Test no path found	True: Pass Test	True: Pass Test

Build Functions Test			
Number	**Test Case**	**Expected Results**	**Actual Results**
3.1	Test build wild card use	True Pass Test	Failed test due to not matching to a string that is 2000+ items long so unable to test against

6.0 Requirement Acceptance Test

Software Requirement No	**Test**	**Implemented (Full /Partial/ None)**	**Test Results (Pass/ Fail)**	**Comments (for partial implementation or failed test results)**
1.The program will read a dictionary text file supplied by the user	Type in dictionary file name with file extensions and see if program runs	Full	Pass	
2.The program will take keyboard only input from user	Mouse clicks do not effect program tested by clicking in terminal	Full	Pass	
3.The program will only make changes to the word one letter at a time	Check all created paths for multiple different inputs	Full	Pass	
4.The program will only allow words that match a list supplied by user called dictionary.txt	Words that are not in dictionary inputted	Partial	Pass	Partial only because the program does not prompt for another input just outputs no path found
5.The program shall only use a word in the path as long as it has not been exclude from the list after user input.	Check to see if words that have been manually inputted into exclude list do not show up in a path they would be present in other wise	Partial	Pass	Words that have been entered into the exclude input prompt are left out from the final word path however it is only a partial implementation as it only allows for one word exclusions not multiple.

6.The program will allow for text input	Attempt to type input	Full	Pass	
7.The program will go from words lead to gold in 3 steps and hide to seek in 6 steps	Test the pre defined paths	Full	Pass	
8.The program will be unit test compliant.	Run Unittest.py file and see if errors are present and if the desired outcomes are present	Partial	Pass	When running the unit test file in PyCharm which we used for version control it says there is no unit test directory. However running it externally on a secondary python interpreter the file run's as intended.
9.The program will be version controlled using GitHub	Check version control updates on GitHub	Full	Pass	
10.The program will only take two words of same length to find a path	Type two separate length words and see if re prompted	Partial	Fail	Attempted to implement however got to a stage where the code would detect a difference in length but would then infinite loop inputs even if same length.
11.The program will only take alphabetical character(no special characters or numbers)	Enter non alphabetic characters and see if code picks up on errors	Full	Pass	

7.0 User Instructions

- Make a clone of the GitHub repository SoftwareTechAssignment from link https://github.com/joshjpm/SoftwareTechAssignment
- Once saved in a local directory run the word_ladder.py file in a python 3 or later environment.
- Follow the on screen prompts and enter “dictionary.txt” as the supplied dictionary
- Enter any words you wish to exclude or just hit enter
- Enter a start word
- Enter a target word
- Look at the steps in which the program is able to take your word from start to target.

8.0 Final GitHub Commits & Version History

Commits on Aug 20, 2017

1. **Update README.md …**

   joshjpm committed on **GitHub** a minute ago

   d295201

2. **Delete Python Word Ladder Debugging and Version Control Assignment 2.……**

   joshjpm committed on **GitHub** 2 minutes ago

   7a800dd

3. **Add files via upload …**

   joshjpm committed on **GitHub** 2 minutes ago

   91e4508

4. **Delete Sample Documentation.doc …**

   joshjpm committed on **GitHub** 3 minutes ago

   af44abe

5.

Delete ~$mple Documentation.doc …

joshjpm committed on **GitHub** 31 minutes ago

8ec3564

6.

Update word_ladder.py …

joshjpm committed on **GitHub** an hour ago

f678f41

7.

Delete Assignment 1.pdf …

joshjpm committed on **GitHub** an hour ago

2ea8bb3

8.

Added in pseudocode, block diagram, additional list functions details.

valkyriak committed an hour ago

a0bea01

9.

Added in pseudocode, block diagram, additional list functions details.

valkyriak committed 2 hours ago

a1ce9d8

10.

Removed unnecessary files.

valkyriak committed 3 hours ago

33681ec

11.

Merge remote-tracking branch 'origin/master'

valkyriak committed 3 hours ago

58da9b8

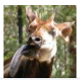
12.

Additional comments added.

valkyriak committed 3 hours ago

e6064f7

Commits on Aug 19, 2017

1. 

Added final documentation file

valkyriak committed on **GitHub** a day ago

c6eac2e

2.

Added in assignment documentation into the directory.

valkyriak committed a day ago

724b465

3.

Added in assignment documentation into the directory.

valkyriak committed a day ago

2d744d1

4.

Merge remote-tracking branch 'origin/master'

valkyriak committed a day ago

d6c489e

5.

Added comments for most functions and important pieces of code. …

valkyriak committed a day ago

2250466

Commits on Aug 18, 2017

1.

added part 7

joshjpm committed 3 days ago

1f0c778

2.

Merge branch 'master' of https://github.com/joshjpm/SoftwareTechAssig……

joshjpm committed 3 days ago

cece0e1

3.

added part 7

joshjpm committed 3 days ago

f1b2502

4.

Update README.md

joshjpm committed on **GitHub** 3 days ago

2bd3ff7

5.

file used for handling word exceptions

joshjpm committed 3 days ago

ba6c0db

6.

finished code. completes lead to gold and hide to seek: …

joshjpm committed 3 days ago

1606ba6

7.

added part 5,6 fin

joshjpm committed 3 days ago

9cfa676

8.

added no match test

joshjpm committed 3 days ago

281d24a

9.

Merge branch 'master' of https://github.com/joshjpm/SoftwareTechAssig… …

joshjpm committed 3 days ago

e839436

Commits on Aug 17, 2017

1.

add 0 match test

joshjpm committed 3 days ago

359d9df

2.

Edited input error statement

valkyriak committed 3 days ago

c5eb796

3.

Edited input error statement

valkyriak committed 3 days ago

99c10f8

4.

added .isalpha code to check only letters in input

joshjpm committed 3 days ago

8dfdc58

5.

part 1,2,3 and half of 4 complete

joshjpm committed 3 days ago

d6fc4ac

6.

added user input handleing for case variants

joshjpm committed 3 days ago

f178568

7.

fixed one of the testings for the find function

joshjpm committed 3 days ago

0a92c45

8.

Also python unittest seems to not want to work with pycharm, however … …

joshjpm committed 3 days ago

da4bfb8

9.

push the unit test file that runs automated testing on the functoins

joshjpm committed 3 days ago

50a3d05

Commits on Aug 16, 2017

1.

changed the printing of path length to match the correct amount.

joshjpm committed 4 days ago

4219e75

2.

Lead to gold now works, "lead > load > goad > gold"

valkyriak committed 4 days ago

93c98eb

3.

Commented a requirement for later correction …

valkyriak committed 4 days ago

048b994

Commits on Aug 15, 2017

1.

Initiated code optimisation, change goes "lead > goad > gold"

valkyriak committed 5 days ago

81e3964

2.

Initiated code optimisation, change goes "lead > goad > gold"

valkyriak committed 5 days ago

66e985e

Commits on Aug 8, 2017

1. **Merge remote-tracking branch 'origin/master' …**

   valkyriak committed 12 days ago

   6b561e0

2. **Kane's Change**

   valkyriak committed 12 days ago

   26130a6

3. **adding a comment**

   joshjpm committed 12 days ago

   b083202

4. **First GitHub Intergration**

   joshjpm committed 12 days ago

   0000949

5.

Add files via upload

joshjpm committed on **GitHub** 12 days ago

f0b7301

6.

Initial commit

joshjpm committed 12 days ago

1cff792