Irfan Shaik
CS314
Professor Ames

Final Project Proposal

As software engineering becomes more relevant in various industries it is becoming increasingly important that the average person have at least a basic understanding of how to code. Learning the basics of programming can be useful to a banker to automate rote data entry and for doctors to be able to use software assisted diagnostic tools. In addition, software tools should also be more easy to use so that a professional programmer can create quick mockups of algorithms and test them. In order to assist the average person to better understand how to code, I created a compiler for pseudocode.

My project is a natural language processing enabled compiler which allows the user to input pseudocode and have it translated to executable Python code. The project will allow a user to input a program such as:

```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p - 1)
        quicksort(A, p + 1, hi)

algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo
    for j := lo to hi - 1 do
        if A[j] < pivot then
            swap A[i] with A[j]
            i := i + 1
    swap A[i] with A[hi]
    return i
```

The output code will be executable without editing and look like:

```
def quicksort(a, lo, hi) :
  if lo<hi   :
    p = partition(a, lo, hi)
    quicksort(a, lo, p - 1)
    quicksort(a, p + 1, hi)

def partition(a, lo, hi) :
  pivot = a[hi]
  i = lo
  for j  = lo to hi - 1  :
    if a[j]<pivot   :
      a[i] , a[j] = a[j] , a[i]
      i = i + 1
  a[i] , a[hi] = a[hi] , a[i]
  return i
```

Irfan Shaik
CS314
Professor Ames

My project is based around understanding Python reserved words, such as function declarations, conditionals, and variables. I will structure my code around finding these Python specific words and then use natural language processing to find words that could map to these language specific keywords given the context.

I will parse the input for keywords words such as ‘def’, ‘if’, ‘else’, ‘then’, ‘next’, and attempt to understand them in the context of the program. For example if def is seen in the code, the following tokens will be interpreted as a function header. If ‘if’ is seen the following tokens can be interpreted to be part of a conditional. If ‘swap’ is seen, there will be a separator token such as ‘with’ or ‘and’; the variables around these separator tokens must be swapped. If we take the statements

```
swap my favorite variable with variable two
switch my favorite variable with variable two
```

The compiler must understand that the user means swap(variable1, variable2). The output from my code is

```
Variableone, variabletwo = variabletwo, variableone
```

My compiler understands the keywords, swap and with. It sees swap as a keyword and with as a delimiter for the first and second arguments. The compiler then combines the relevant words to create variables and outputs the correct syntax code. I created the following grammar to help me identify the variables in the sentence:

```
<SWAP> <ARGUMENT ONE> <DELIMITER> <ARGUMENT TWO>
```

I use word2vec in Python to read a corpus of pseudocode and identify which words could be a command, meaning they have a strong association with |SWAP| and which words are delimiters meaning they have a strong association to |DELIMITER|. The rest of the arguments must be the variables. Other words in my grammar are:

```
<EQUALITY> <ARGUMENT ONE> <DELIMITER> <ARGUMENT TWO>
<CONDITIONAL> <ARGUMENT ONE>
<DISPLAY> <ARGUMENT ONE>
<LOOP>
```

Due to the difficulty of my project, I expect the code to do a perfect translation to Python in a few test cases and need the user to correct the syntax issues in the output in most test cases. However, if the user uses the pseudocode grammar that I provide, the output will be perfect in all test cases.