Diploma in Computer Science Project Progress Report

# Exploration of Spoken Programming Languages

January 29, 2019

**Name:** James Hargreaves **College:** Gonville and Caius **crsid:** jh2045

**Project Supervisor:** Paula Buttery

**Director of Studies:** Graham Titmus

**Overseers:** Robert Mullins and Pietro Lio

## 1 Introduction:

In this project I am comparing different approaches to build an application that takes a spoken language transcript and maps it to pseudocode. The transcripts are of a programmer explaining a function in the most natural way to them.

## 2 Checklist

- ☑ Data Collection.
- ☑ Write module to remove variable and function names.
- ☑ Write Baseline module.
- ☑ Write module which uses Statistical Machine Translation.
- ☐ Write module which uses Traditional Machine Translation Techniques.
- ☑ Write Evaluation module.

## 3 Work Completed:

1. Collect Corpus - I wrote a web app which present the respondent with a prompt for a simple algorithm and recorded their spoken response. I got 14 respondents giving me a data set of size 135. This data was then transcribed into a text file using the app at trint.com and then manually checking / correcting mistakes. I then defined a pseudocode grammar and for each transcription I typed out the pseudocode which I felt best matched it.

2. Entity recognition - This first thing I did was build python scripts to convert the two text files to an intermediate representation by replacing variables and function names with a variable token. This step also removed the bracketing from the pseudocode representation.

3. Mapping spoken phrases to pseudocode tokens - I produced a list of mappings from English phrases to pseudocode tokens using the list at https://blog.codinghorror.com/ascii-pronunciation-rules-for-programmers/ . After looking at the list I manually added some of my own e.g. '*' to 'multiplied by'. Each transcript was processed to a stream of pseudocode tokens using longest prefix matching.

4. Baseline - Using the output stream mentioned in the above point, I trained and used an n-gram model to compute the most likely ordering of the given tokens.

5. Statistical Machine Translation - built word alignment models using IBM models 1 and 2 and use these to build phrase alignment tables. This combined with a n-gram model was used to construct a beam search decoder. I also experimented with pruning.

6. Evaluation - i wrote a python script which takes a translation produced by the system and the actual pseudocode and computes the minimum edit distance.

# 4 Still To Do:

1. I still need to construct the module that uses traditional rule based machine translation techniques.

# 5 Timetable:

The work I have completed covers is everything I planned to have done by the 3rd of February other than the tradition machine translation module. However I had planned in a 2 week overflow time from the 4th of February and so I am not concerned about being behind.

Word count = 433