AINT512 Project Proposal

# Abstract:

This report outlines a text-adventure based conversational agent, which uses speech instead of monitors and keyboards. Section 1 explains the projects design with justification; section 2 details the design method for the conversational agent; section 3 discusses the results and feedback from play testers; and section 4 summarises the project.

# 1 – Design Overview

## 1.1 – Game Design

To allow for rapid development of the game, along with various game features, a simple game-engine for text adventures was developed. This enabled rapid feedback integration, but also meant certain features became too complex to develop, such as being able to pause or interrupt the narration.

A pre-existing game, ‘The Stanley Parable’, was used as a guide for the design of the computer’s personality, and the general flow of the game. The games layout was designed using tools from the choose-your-own-adventure genre of novels; the story was created from personal experiences; and the tutorial designed iteratively via numerous playtests.

## 1.2 – Speech To Text Design

Grammars were originally used for the speech to text algorithm, as parsing several grammars allows for a direct connection between an expected sentence and the desired action of the user. However, due to the wide variety of possible input phrases a user could generate for a given scenario in game, grammars were deemed infeasible. To generate a suitable textual corpus, a grammar of synonyms was generated for each word. Inflexions, such as ‘umm’, were inserted between each word as a conditional. The number of possibilities that needed to be scanned by the grammar amounted to several minutes’ worth of processing; an unreasonable amount of time for any user to be expected to wait.

Statistical language models were tested as a substitute. Specifically, CMU-Sphinx was tested as an alternative offline model. CMU-Sphinx also allowed grammars to be tested, and could use custom dictionaries. This allowed for direct comparisons with grammars, and improved processing speeds as a limited dictionary size could be used. The results were usable, but highly dependent on a user’s pronunciation and accent. To alleviate this issue, two online language models are also used in conjunction, Google and Wit.

Each of the three language models process user inputs into textual results, along with an associated confidence metric. These results are treated as bags-of-words, where each word is assigned a weight towards a specific narrative option. The weights are calculated via the Term Frequency Inverse Document Frequency (TFIDF) metric. The results were multiplied by the confidence of each result, and all of those summed together. These metrics, one for each narrative selection, are then sorted from largest to smallest. If the largest metric is not sufficiently large enough, then it is likely that a user attempted to select an invalid option. If the two largest metrics are of similar value, then a selection cannot be made. This is likely an issue relating to word weights, or how the user was guided toward that answer. Otherwise, the selection is considered valid.

# 2 – Design Process:

## 2.1 – Basics:

***Task Domain:*** The program domain is to present and guide a user through a custom-made choose your own adventure story.  
***Dialogue Specification:*** The dialogue specification, or narrative paths, can be seen in ***figure x***.  
***Personality Specification:*** The narration used is polite, well spoken, and slightly humorous.

## 2.2 – Grammar Design:

User inputs are expected to be in one of several formats. Yes and no answers either acknowledge or reject the directed narrative. Direct responses, containing a single word or a short phrase, will directly mention some aspect of the directed narrative. This will provide enough context to infer the user’s selection. Due to using language models and the bag-of-words approach, indirect responses should react similarly to the direct responses.

## 2.3 – Dialogue Design:

The dialogue, or narrative, is built up using a variety of techniques. Specifically scenes, sequels, directed dialogues, user input and computer processing cues, and the three standard options for narratives in choose your own adventures.

Scenes are moments of conflict or hardship, experiences by both the character and the reader. Sequels present the character’s reaction to the previous scene, and set up motivations for the next scene. Directed dialogues guide the user towards available narrative options without having to list them. User input and computer processing cues alert the user to internal states of the computer, allowing them to interact at valid times without being frustrated at slow responses. Finally, the three standard options are ‘yes’, following the directed narrative, ‘no’, doing the exact opposite of the directed narrative, and ‘something else’, which are alternative options that a user might try in such a scenario. These alternatives may often be equivalent to asking ‘what else can I do’, such as saying ‘look around’. Deducing valid alternatives comes down to narrative choice and play-tester feedback.

## 2.4 – Error Catching & Handling:

The program distinguishes between four types of errors. The simplest of these is silence, whereby no input is provided by the user. Under this condition, the program displays a note of the silence and continues to wait for user input again. Any other response would interfere with the user’s immersion within the game.

Another error type results from non-vocal, or unintelligible, inputs. Again, to prevent a break in immersion, these errors are simply ignored. However, should the confidence of a translation from such inputs be high enough, the error will become a mistranslation. Incorrect responses would also fall under this category. These errors are difficult to detect due to the nature of language models and the bag-of-words algorithm, and is further compounded with the addition of synonyms for each narrative option. Should no valid selection be identified, a user error is raised, and the user is kindly asked to try again in a variety of different methods.

The final error occurs when multiple narrative options are identified given the users input. This is considered a creator error as most inputs will be valid, but poor word weightings within the program prevented a correct assignment. The program humorously apologises and asks the user to rephrase their input in an attempt to prevent the error occurring again.

## 2.5 – Action / Response Generation:

Responses are entirely pre-scripted. Certain sections can change based on internal states within the program, such as ‘how many times the user has been to this part of the narrative before’. The only exception to this is the user error, which can help in one of two ways. These are either confirmation of the computer mishearing the user, or verification that the computer heard correctly, but the desired path was invalid.

## 2.6 – User Evaluations:

Three evaluation metrics have been generated. The first is a measure of how successfully the program infers the correct narrative selection based on the users input. The second metric categorises user inputs, and the third provides the likelihood of a user entering a particular section of the narrative.

The program can make four possible inference types using two parameters. If the translation is a somewhat accurate representation of what the user actually said, then it is considered ‘true’, otherwise, it is considered ‘false’. If the program selects the intended narrative option it is considered ‘correct’, otherwise, it is considered ‘incorrect’.

User inputs are categorised into three types: a ‘yes’ / ‘no’ type if the users inputs roughly translate to such simple statements; a sentence-based director type, where the user explicitly instructs the character to do something; and finally, a sentence-based actor type, where the user could be immersing themselves into the game.

The third metric is a simple histogram over the different sections of the narrative covered.