Story Generation-Creation

Liam O' Brien

16416736

Final Year Project 2020

B.Sc. Single Honours in Computer Science

Maynooth University
National University of Ireland Maynooth

Department of Computer Science

Maynooth University

Maynooth, Co. Kildare

Ireland

Supervisor: *Dr. Diarmuid O' Donoghue*

Declaration

I declare that this thesis, which I now submit for assessment is the result of my own independent work and investigation, except where otherwise stated. I give consent for this material to made available for distribution to future final year students as an example of the standard required for final year projects.

Signed: LIAM O' BRIEN Date: 11th April 2020.

Acknowledgements

I would like to thank my supervisor Dr. Diarmuid O' Donoghue for giving me the opportunity to undertake this project and for providing continual guidance and support through the process. I would also like to thank my friends and family for reviewing this thesis before submission.

Abstract

This paper describes the creation of infrastructure with potential uses in story generation and memory enhancement testing. There were two main focuses for the infrastructure developed. One was in finding semantic connections between nouns and creating sentences between them. The other area of work focuses on mapping nouns onto a list of visualisable nouns. Software was created using the Python technology to achieve these goals. In particular, it was shown to be possible to put a randomly ordered set of words into a coherent sequence. Additionally, the software created allowed for a list of surnames to be mapped to visualisable nouns, with the list of mappings sorted into a coherent sequence. A story was generated from this sequence by generating a sentence between each word pair in the sequence. It is the belief of the writer that the infrastructure developed could be used in further studies related to software aiding in the retention of information.

Table of Contents

Chapter One: Introduction & Project Overview

There are many techniques used to help a person retain information. Two among them are the use of stories and a mnemonic technique called *memory palace.* Using a story to remember a group of words involves chaining the words together by creating simple and easy to remember sentences between them. With *memory palace*, a list of words is remembered by representing them as simple and easy to remember objects. These objects are then pictured as being in familiar locations.

The concept of a Story Generator Algorithm (SGA) is core to this paper. Simply put, an SGA is a sequence of steps in which a list of inputted words is used to construct a story. Much research has already been done on SGAs, with notable works including Sheldon Klein's *Novel Writer*, James Meehan's *TaleSpin* and Péréz v Péréz's *MEXICA.*

Novel Writer was famed for its ability to generate "2100 word murder mystery stores". *TaleSpin* was capable of generating stories about woodland creatures through a process of problem solving and planning by software. *MEXICA* is a modern SGA which is still being researched. The SGA creates stories by alternating between two phases of processing.

Another important term is that of a visualisable noun. This is a word that can be easily pictured and remembered. The list of visualisable objects used in this work are all nouns that come from a library of 3D objects. This is to allow for further applications of the software in VR or AR systems.

There are two main areas of work to be done. One area will focus on finding semantic connections between nouns. These connections are to be used to create simple sentences. By chaining together these simple sentences a story can be generated. In order for the story generated to be as coherent as possible, it is imperative that the list of nouns presented to a constructed SGA are in a coherent order themselves. So, a sorting algorithm must also be developed which sorts a list of nouns into some coherent order.

The other area of work will be in mapping a presented noun to a visualisable object. Software that performs this mapping would allow for non-concrete nouns (such as surnames, places, etc.) to be represented by simpler nouns. These simpler nouns could then be used in sentence generation in place of the more complicated noun.

The writer aims to use the Python language to create the desired functionality. This will allow for the use lists and dictionaries to easily store and read data, as well as ease the development process. Technologies such as ConceptNet & WordNet will be used to identify semantic connections between concepts. The SimpleNLG technology will be used to simplify the process of creating sentences.

The rest of the paper is organised as follows. Chapter two contains information about ground-breaking SGAs, as well as an overview of ConceptNet and WordNet, two technologies to be used by the software developed. Chapter three details the specific problems to be solved and the approaches taken to solve these problems. Chapter four gives examples of the software in action, as well as evaluating the software to be constructed. Finally, chapter five contains the main conclusions of the paper and suggests future work that could be taken.

Chapter Two: Literature Review & Background

A Story Generator Algorithm (SGA) is simply a sequence of computational steps that produces a story as its end-product. There are many techniques used in SGAs, with some of the most frequent being the use of planning and the abstraction of key elements in a story to match a particular template. The field of research into SGAs is one with a rich history, but the term itself is quiet recent. Indeed, the concept first appeared "in 2004 as the name of a project of the Hamburg Narratology Research Group"[1].

2.1 Joseph E. Grimes' Fairy Tale Generator

In the field of computational creativity, it was until recently accepted that Sheldon Klein's *Novel Writer* system was the earliest algorithmic story generator designed. However, a paper[2] by James Ryan details a story generator created by Joseph E. Grimes in the 1960s, predating Klein's 1971 novel writer . Grimes initially programmed an IBM 650 computer to tell stories following a basic fairy tale pattern. Stories were constructed using a Monte Carlo approach, with actions taken, character roles and the narrative path chosen at random.

2.2 Sheldon Klein's Novel Writer

Sheldon Klein's 1973 *Novel Writer* system was programmed in FORTRAN V on a Univac 1108. The system was reported as capable of generating "2100 word murder mystery stories, complete with semantic deep structure, in less than 19 seconds"[3]. A directed graph with labelled edges (where nodes are semantic objects and labelled edges are relations connecting the objects) was used to represent the modelled universe of the story. The narrative structure of the story was generated by a micro-simulation model, which was written in compiler-driven simulation language.

The novel writer was capable of creating murder stories in a weekend party setting. Specification of character traits and a description of the world were given as inputs to the novel writer. The deterministic aspects of the algorithm guaranteed a murder story, but the motives and particular murderer/victim varied stochastically. In particular, the motives for murder arose as a function of events during the generation of the story. The flow of the story was a result of successive reports on the changing state of the modelled universe.

The novel writer was a two part algorithm. The first part consists of a highly constraining set of rules that encode possible changes in the world state, while the second part details a sequence of scenes detailing the type of story to be told. These constraining rules meant that while many stories could be generated by the algorithm, the changes between them consisted only of differences in who the murderer was, their motive, the victim, and other such details.

2.3 James Meehan's Tale-Spin (1976)

James Meehan's Tale-Spin was another early story generator which pioneered the generation of stories via carefully-crafted processes over assembling stories from pre-defined texts (as in a Choose Your Own Adventure book). Tale-Spin generated stories about woodland creatures, with a focus on simulating character reasoning and behaviour. A key element of Tale-Spin is a world model that contains humans, talking animals, motivations for actions and the physical world (mountains, caves, forests).

In his thesis Meehan states "at the heart of Tale-Spin is a problem solver"[4]. Planning plays a key part in generating stories. The stories generated by Tale-Spin can be considered as accounts of what happened during the course of solving problems. Indeed, stories are generated from the interaction between three active components in software; a problem solver, an assertion mechanism and an inference mechanism.

2.4 Pérez y Pérez's MEXICA (1999-present)

MEXICA[5] is a computer model used to generate stories about early inhabitants of Mexico. The model contains two phases: the engagement phase and the reflective phase. In the engagement phase, new story material is progressively generated. In the reflection phase generated material is revised to ensure that constraints are met. Pérez designed *MEXICA* in terms of two main processes, one which creates data-structures in long term memory and another which creates new stories based on the data structures.

MEXICA creates stories by cycling between the engaged and reflective states. While engaged, actions are performed which produce context in the story world. This context is compared against other stored contexts in data structures in memory to find similar situations. The data structures give a set of possible next actions which are then loaded into working memory. *MEXICA* chooses an action to take, which produces new story-world context and thus continues the cycle. If the cycle of engagement is broken, the system enters into the reflective state. In this state all preconditions are verified and impasses broken.

What sets *MEXICA* apart is that the system uses emotional links and tensions between characters as a driver for generating new material. Emotional links are represented by percentage tables, which contain the value of intensity of emotional links between characters. Tensions in the system may be triggered by postconditions or inferred postconditions.

2.5 Method of Loci

The method of loci, commonly referred to as “memory palace”, is a strategy of visualising concepts to be remembered at specific locations along a route. The purpose of this is that as the person imagines themselves walking along the route they can picture the concepts to be remembered.

For example, in memorising a list of chemicals a person could first try to represent the chemicals by simpler words. So hydrogen could be represented by the sun, beryllium by a berry and so forth. Then the easier to remember concepts could be placed along a familiar route. E.g. the sun could be a the bottom of the stairs, a berry might be found on a landing on the staircase etc.

The use of method of loci has long been used for memory based applications, and has been shown to improve performance in such applications. Indeed, it has been stated that “the use of MOL leads to better memory and recall”, as was demonstrated in an article published on ResearchGate[6].

2.6 Using Stories to Aid Memory

The use of storytelling techniques to improve retention of information is one with precedent. This method is commonly referred to as a mnemonic link system. The process involves linking together concepts with a verb to form a simple sentence. It is believed that the human mind has a natural ability to remember stories, so by remembering a story constructed from a list of words it is theorised that memorising the initial list of words becomes easier. Indeed, it was shown in a dissertation paper by Tommy Oaks from the University of Tennessee[7] that students that received education from a storytelling method performed better than students which were educated in a traditional manner.

2.7 ConceptNet

ConceptNet[8] is a semantic network which aims to help computers understand the meanings of words. The network is expressed as a directed graph. The nodes of the graph are concepts and the edges connecting them represent the relationships between concepts. Concepts in *ConceptNet* represent noun phrases, verb phrases, adjective phrases and clauses. The relationship represented in *ConceptNet* are seen as "common sense" relations, such as "AtLocation", "HasContext", "IsA", etc.

2.8 WordNet

WordNet[9] is a lexical database containing semantic relations between words. *WordNet* groups words into sets of semantic relations. These sets included sets of synonyms (synsets), sets of hyponyms and sets of meronyms. *WordNet* resembles a thesaurus in the regard that it groups words together based on their meanings. However, unlike a thesaurus, *WordNet* interlinks specific senses of words. This allows *WordNet* to disambiguate words when grouping.

2.9 SimpleNLG

The *SimpleNLG*[10] technology allows for simple sentence realisation but also boasts features such as setting the tense of sentences, allowing for multiple clauses, allowing the addition of complements or modifiers, and offering the ability to create interrogative sentences.

Setting the tense of the sentence to "Tense.PAST" allows for sentences such as "The girl was in the school." Similarly, specifying the feature "Tense.FUTURE" allows for sentences such as "The girl will be in the school." The default tense considered by *SimpleNLG* is the present tense, which is primarily used by the software developed in this project.

Modifiers in *SimpleNLG* can be considered as adjectives. They can be added to a sentence by creating noun phrases and using the addModifier method to attach adjectives to objects/subjects in the sentence.

Chapter Three: Problem Specification & Solution

3.1 Problem Statement

There are two focuses on work in this project. The first area of research is to order a list of inputted nouns in a coherent matter. The ordering of the words should allow for the creation of stories from the nouns. Work on this part of the project will involve finding semantic connections between concepts. From these connections methods are to be developed that can measure the connectivity between words. This will allow for the sorting of a list of inputted words. Finally, methods must be created that allow for sentences to be generated between word pairs.

The other area of work involves mapping nouns onto a set of "visualisable objects". These objects come from a library of 3D objects. Approaches should be developed to take advantage of phonetic similarity and semantic connections.

3.2 Data used

ConceptNet file:

A basis for knowledge mining used in this project was an excel file containing information about two words and a relation between them. The file contains a list of 919853 rows with three columns. The first column of each row contains the subject of the relation, the second column the object of the relation and the third column the actual relation.

Examples of rows in the file are given below.

wheel	steering	UsedFor
girl	school	AtLocation
gun	police	HasContext
dog	animal	IsA

The above can be read as follows. The *wheel* is *UsedFor* steeting. The *girl* is *AtLocation school*. The *gun HasContext police*. The *dog IsA animal*.

Representing the data

It was decided to represent the data in the excel file as a Python dictionary. The reasons behind this were that dictionaries in Python boast quick access speed, which is vital for representing a dataset the size of the ConceptNet file. The key of the dictionary was the first word, and the value was the second word and an encoded representation of the relation between them.

Accessing the dictionary

The dictionary is accessible by indexing the dictionary with a word contained in it. For example, indexing the dictionary with “wheel”, i.e. CNdict[“wheel”] returns a list of words related to wheel, and the relationship between them. Examples of elements in said list are […, steering47, rolling47, …]. The meaning of the returned list can be interpreted as wheel being related to steering as a wheel is *used for* steering.

3.3 Mapping to Visualisable Objects:

Multiple approaches were taken in mapping a noun to a visualisable object. Chief among them is mapping a noun to a visualisable object that it has the greatest phonetic similarity to. So for example, the word “broth” might have been found to have the greatest phonetic similarity to “bath”, so “bath” was considered a potential mapping. The pyphonetics library was used to calculate phonetic similarity.

Another approach was to use WordNet to map an inputted noun to any visualisable object that it shares a semantic meaning with. WordNet boasts synsets, which are sets of words that have the same sematic meaning in some context. E.g. if “cur” is to be mapped, this approach would discover that “cur” has the same semantic meaning as “dog” in some context (i.e. “dog” is in the synset of “cur”). So “dog” could be considered a potential mapping.

The last significant approach taken was to map a noun to a visualisable object that it has numerous direct or indirect connections to. This was achieved using the software constructed in Section 3.4. For example, the software could discover that “orange” is directly connected to “apple” and shares numerous indirect connections. E.g. “apple” and “orange” are both connected to “tree”, so they’re indirectly connected via “tree”. Due to these direct and indirect connections, “apple” might be considered as a potential mapping for “orange”.

The technology developed for mapping words can be seen in use in Section 4.1, where surnames were mapped to visualisable objects.

3.4 Sorting the words

Finding connectivity between words

An important concept in this work is the idea of connectivity. Concepts in ConceptNet are connected if they are either directly connected or indirectly connected. If two concepts are directly connected then there exists a relation between them. For example, “school” and “class” are directly connected because “class” is related to “school” via the relation “AtLocation”. In other words, the class is at the school.

Concepts in ConceptNet can also be indirectly connected. This means that there exists of series of relations between the two concepts that connect them. For example, “tree” is indirectly related to “bird” as “bird” is “AtLocation” “nest”, and “nest” is “AtLocation” “tree”.

Determining if two concepts are directly connected is a trivial matter of accessing the ConceptNet dictionary with either word as a key and checking if the other word is contained in the list of returned values.

The process for finding indirect connections takes more work and is described below:

For all words directly connected to word A

Check if the word is directly connected to B, if it is store the word in a list.

The process is repeated with concepts A and B swapping positions. Finding indirect connections of greater than one connecting word involves looping through all direct connections of all direct connections of a word, i.e. a new loop must be introduced to find another word in-between.

Measuring how related words are

For the purpose of ordering a list of given concepts it was of importance to create a measure of how closely related two words are. This would allow for a sorting algorithm to sort the list of words such that the adjacent elements to the word are most closely related to it.

To measure how closely related words are, a metric called the Rscore was devised. The metric is a weighted sum of the number of indirect relations between two concepts, as well as an additional score that is added if the words are directly connected.

The metric was devised so that words that a person might consider closely related, e.g. “guitar” and “guitarist”, have a high Rscore. Words that might be considered unrelated by a person, e.g. “bookcase” and “field”, would have a lower Rscore.

Section 4.4 shows the effectiveness of using the Rscore metric in a sorting algorithm for a list of words.

Sorting the words

An overarching goal of this project was to sort a given list of concepts in such a manner that adjacent entries in the final list are as closely related as possible. The approach taken was to select one word from the inputted list (the first word was taken as convention) and append it to a new list. From the pool of other words not selected yet, the word mostly closely related to the first word, based on the Rscore metric, was appended to the list and removed from the pool of available words. This process continued until all words were selected.

The sorting algorithm devised can be seen in use in Section 4.1. A look at the effectiveness of using a sorting algorithm can be seen in Section 4.4.

3.5 Creating a story

For the purpose of this work, a story was simply considered to be a sequence of sentences. The process of creating a story involved reading in input, sorting the input using devised methods and creating a sentence between all adjacent word pairs in the ordered list. The sentences generated were to be short and easily memorable, so an approach was taken to create simple sentences that link concepts together with a verb.

Finding suitable verb

The first step in the process was to create a dictionary containing mappings of ConceptNet relations to verbs. This would allow for suitable verbs to be chosen between related concepts, which could potentially make the sentence easier to remember. Software was created that accessed the dictionary containing these mappings and returned an appropriate number of verbs. For example, the relation “IsA” corresponded to the verb “is”, the relation “AtLocation” corresponded to the verb “is in”, etc.

For directly connected concepts, the dictionary was indexed with the relation between the concepts and a single verb was returned. For example, “cat” has the relation “IsA” with “animal”. When the dictionary is indexed with “IsA”, the verb “is” is returned.

The process becomes more complicated when indirect connections are considered. An approach was taken to return at most two verbs for the purposes of story generation, as it was believed than any more would result in an overlong sentence.

So for indirectly connected concepts with one adjoining concept, two verbs were turned. These verbs represent the relation each word has to their shared connection. For example, “bag” is indirectly connected to “baby” as both concepts are connected to “car” via the relation “AtLocation”. So the two verbs returned could be “is at” and “is at”.

Constructing a sentence

Based on the type of connection between two concepts, different approaches were taken to construct a sentence.

For directly connected concepts, constructing a sentence was a simple matter of finding a suitable verb that connects the concepts and determining whether a concept was an object or a subject. A concept was considered an object if it was the concept acted upon. For example in the sentence “The boy wears glasses”, the object is “glasses”. The subject is the concept that is performing the action, which would be the “boy” in the example.

For two concepts A and B, if the concept A was contained in the set of elements directly related to B (i.e. in the ConceptNet dictionary with key B) then the object of the sentence was said to be A, and the subject was B and vice versa.

With the verb found and the object and subject set, the sentence was constructed using the SimpleNLG technology.

For example, the concepts “leaf” and “tree” are related in ConceptNet by “leaf” being “AtLocation” “tree”. The object of this relation is “tree”, and the subject is “leaf”. The verb to connect these concepts as decided by previously developed methods is “desires”. Ergo, the sentence generated from the software is “The leaf is in the tree.”

Indirect connections pose the problem of more than one verb being necessary to generate a sentence. For the case of concepts being indirectly connected via one word, two cases were considered for sentence generation.

If the two concepts share the same relationship to the word that connects them, then a sentence could be created by considering the two concepts as subjects and their shared relation as the object of the sentence. The verb in the sentence was found using already developed methods.

For example, the concepts “boy” and “girl” are indirectly connected by both being “AtLocation” “school”. “Boy” and “girl” are considered the subjects of the sentence, the “school” is the object and the verb was “in”. Thus the sentence generated would be “the boy and the girl were in the school”.

For the case of the two concepts having dissimilar relationships to their connecting word, a different approach was needed. Instead of finding a single verb to use, two verbs were to be found. To do this, each concept’s relation to the connected word was considered, and a verb was found for each relationship. The sentence was split into two clauses and connected with the conjunction “and”. For example, the concepts “bird” and “water” were found to be indirectly connected via the concept “sea”. However, “bird” was related to “sea” via “AtLocation” and “water” was related to “sea” via “partOf”. The sentence generated from these relations was “the bird is in the sea and the water is a part of the sea.”

The approach of creating multiple clauses was not taken for indirect connections with greater than one connected word. This decision was made to avoid overly long sentences that would not aid in helping a user remember words. When no direct connections or indirect connections with one connecting word were found, the approach taken was to select a random relation from one concept and any of the words its directly connected to and extend that relationship to the other concept. For example, the concepts “nail” and “field of study” are not directly connected and have no indirect connections with fewer than two connecting words. A randomly selected connected concept to “nail” is “tool” and the relation is “typeOf”. This relationship was extended to “field of study” and the sentence generated was “the nail and the field of study are a kind of tool.”

Alternative presentation

After development of a story generator was completed, an issue was identified with the presentation of the sentences. As the goal of the infrastructure is to be used in memory enhancement tests, the sentences should be concise, and the words to be remembered should be clearly identified. The above presentations were deemed to be unclear and overly long, so an alternative was developed. The new approach was to have a consistent word order amongst the generated stories, with the two relevant words place at the start and the end of the sentence.

An example of this approach is given in Section 4.1.

Chapter Four: Results

4.1 Proof of Concept

There were two overarching goals of the project. One was to use semantic connections between words to create sentences, and another was to perform mappings of nouns onto a list of visualisable objects. To illustrate the achievement of these goals, the following was performed. From a list containing the surnames of Nobel prize winners, ten surnames were selected. These surnames were then mapped to visualisable objects. The mappings were then sorted into a coherent ordering. A story was then constructed by forming sentences between word pairs in the ordered list.

Nobel Prize Winners

An example list of mappings and story generator is provided below.

Names are:
['Karl Manne Georg Siegbahn', 'Aung San Suu Kyi', 'Prince Louis-Victor Pierre Raymond de Broglie', 'Ernst Ruska', 'Gabriel Lippmann', 'Wolfgang Pauli', 'Kim Dae-jung', 'Donald Arthur Glaser', 'Yasser Arafat', 'Hans Albrecht Bethe']

Mappings are:
Karl Manne Georg Siegbahn -> curtain
Aung San Suu Kyi -> bee
Prince Louis-Victor Pierre Raymond de Broglie -> baby
Ernst Ruska -> basin
Gabriel Lippmann -> library
Wolfgang Pauli -> apple
Kim Dae-jung -> dog
Donald Arthur Glaser -> glove
Yasser Arafat -> army
Hans Albrecht Bethe -> bag

List of mappings:
[“curtain”, “bee”, “baby”, “basin”, “library”, “apple”, “dog”, “glove”, “army”, “bag”]

Software sorted mappings:
[“curtain”, “dog”, “bee”, “library”, “apple”, “bag”, “baby”, “basin”, “army”, “glove”]

Story generated:

The curtain and the dog are contextualised by derogatory.
The dog and the bee are in the house.
The bee and the library are in the canada.
The library and the apple are in the house.
The apple and the bag are contextualised by baseball.
The bag and the baby are in the backseat of car.
The baby and the basin are in the cradle.
The basin and the army are a kind of vessel.
The army and the glove are capable of fight enemy.

Clearer presentation

As discussed in Chapter 3, the above story presentation does not lend itself well to memorising the necessary words. Using the alternative presentation as previously mentioned with the same inputted surnames, the story generated is as follows:

The curtain has context derogatory. As does dog.
The dog is in backyard. As is bee.
The bee is in house. As is library.
The library is in house. As is apple.
The apple is in grocery store. As is bag.
The bag is in backseat of car. As is baby.
The baby desires comfort. As does basin.
The basin is a kind of vessel. As is army.
The army is in military base. As is glove.

4.2 Expanding Relations

To demonstrate the possible use of the software developed in the course of this work in regards to finding connections between concepts in a text, methods were developed to identify and suggest new relations from excel files listing relations already identified in a text.

The software developed first reads in an excel file detailing relations between concepts already found in a text. The excel files contain three columns, the first and third column containing two concepts and the middle column the relation between them. The methods developed look for connections between the two concepts in the excel files, and if new connections are found they are added to a list of suggestions which is printed to the screen.

The software was tested on seven excels files, with each file containing 43.286 +/- 14.784 existing relations. The software returned 7.286 +/- 3.039 new suggestions.

4.3 Comparisons of sorting techniques against human sorted input

To gather the user data, four lists of concepts were generated. Two of these lists contained randomly chosen visualisable objects. One lists contained a list of words that were deemed "well connected", i.e. a random visualisable word was chosen and a word directly connected to it was added to the this. This continued until 10 words were selected. The last list contained words chosen specifically by the writer. The list of words can be found in the appendix.

All four lists were randomly shuffled and presented to the user. The user was asked to sort the list of words in a manner that to them would constitute a "good" story. This data was collected and processed for analysis.

The root mean square (RMS) error was considered for analysis. A high RMS value is an indicator that two lists are dissimilar, whereas a low RMS value would indicate similar lists. Two identical lists would have an RMS value of 0.

To demonstrate the effectiveness of the infrastructure created in this work, the software was tested directly against randomness. To do this, the following process was performed.

For each list that was presented to users

Generate a random sorting of the words in the list, call the sorting A

Generate a random sorting of the words in the list and then sort the list using sorting algorithm defined in Section 3.4. Call the sorted list B.

Calculate the RMS value between A and how each user sorted the same list of words. Do the same for list B. Store the RMS values.

Repeating this process 20 times lead to the following results:

User	List	Mean RMS software (B)	Std dev software	Mean RMS random (A)	Std dev random
1	1	30.62	6.27	39.84	6.85
1	2	30.83	11.35	41.69	5.96
1	3	38.15	6.47	38.13	4.61
1	4	22.19	6.99	39.88	5.68
2	1	30.08	5.46	37.86	9.15
2	2	32.44	12.85	39.23	6.48
2	3	49.14	2.18	38.33	6.87
2	4	27.78	6.03	40.83	5.55

Hypothesis test of difference in RMS values

The null hypothesis, H_0 = difference between mean RMS software value and mean RMS random value is zero. The alternative hypothesis, H_A is that the difference is non-zero.

Group 1 was considered to be the mean RMS software values, group 2 was considered the mean RMS random values. The mean of group 1 minus group 2 was calculated to be -6.82. The 95% confidence interval of the difference is (-12.98, -0.65).

With a p-value of 0.0325 and at 95% confidence it can be concluded that difference between mean RMS values of software and random sorted lists is statistically significant.

In other words, there is less error for the software sorted lists than there is for the randomly sorted lists. This could be perceived as meaning that the software sorts the list of words in a manner similar to how a person would.

Analysis of lists used

Looking at the mean RMS values obtained, it would seem that the sorting algorithm developed performed best with list 4. This list was generated by randomly selecting ten words from the list of visualisable objects.

The list of words was: [“brain”, “basket”, “house”, “foot”, “arm”, “sock”, “foot”, “key”, “ball”, “berry”, “seed”].

A potential factor that contributed to the software sorting the list in a similar manner to humans might be that a lot of the words in the list are connected to each other. E.g. “foot” is connected to “arm”, “sock”, “brain” etc. The software would take advantage of this and a human would find it easy to connect these concepts.

Looking at the list of mean RMS values obtained again, it would seem that the software performed the worst with list 3. This was another list that was generated by randomly selecting ten visualisable objects. However this time there seems to be much fewer connections between the words in the list.

The list of words was: [“stocking”, “brush”, “feather”, “fish”, “card”, “chain”, “line”, “key”, “wing”, “engine”].

From looking at the mean RMS values obtained from these lists, it would appear that the software sorts lists more similarly to a person when the list contains “well-connected” words. i.e. when there are multiple connections between words in the list.

4.4 Comparison of word connectivity between random and software sortings

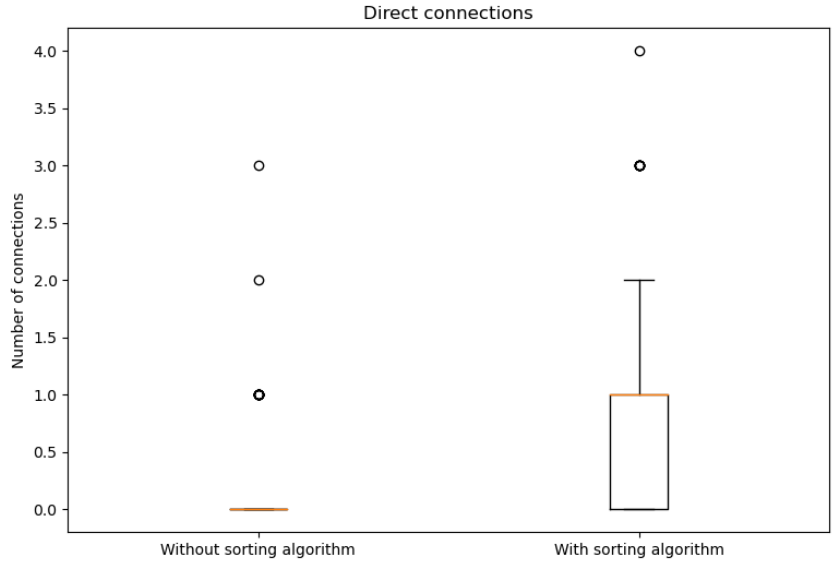
Input is a list of ten randomly selected visualisable objects. The randomly generated list is then ordered using the sorting algorithm developed. Then the number of direct connections, indirect connections and other connections (indirect connections with more than one connecting word) were counted. This process was repeated 100 times and the results obtained are as follows:

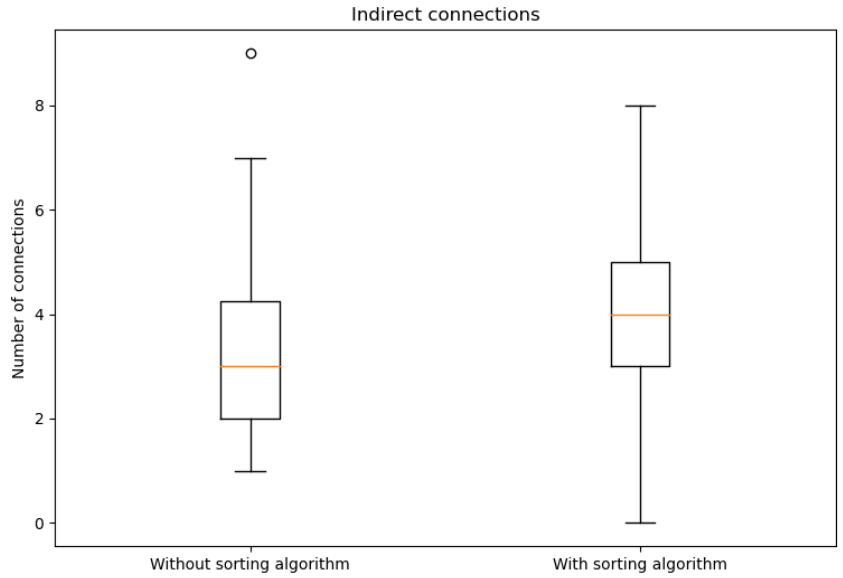
Connection type	Mean number of connections	Standard deviation
Direct	0.87	0.928
Indirect	4.23	1.835
Other	3.9	1.743

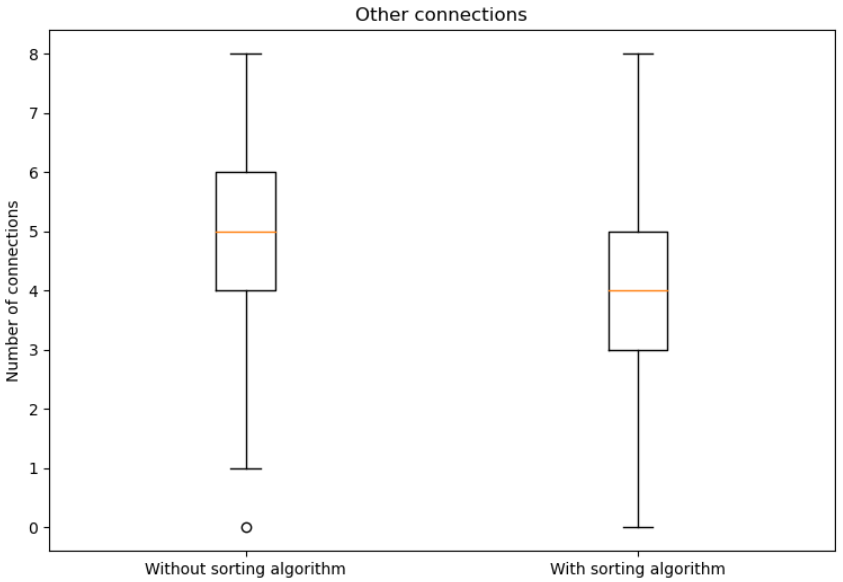
Repeating the same procedure but without putting the input through the sorting algorithm, the following results were obtained.

Connection type	Mean number of connections	Standard deviation
Direct	0.2	0.492
Indirect	3.51	1.605
Other	5.29	1.604

Boxplots of the data are provided overleaf. The boxplots suggest that that the distribution of connection types differs between software sorted and non-software sorted lists. i.e. there appears to be a greater number of direct and indirect connections for the software sorted lists as opposed to randomly sorted lists.


Direct connections
Number of connections
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Without sorting algorithm
With sorting algorithm


Indirect connections
Number of connections
8
6
4
2
0
Without sorting algorithm
With sorting algorithm


Other connections
Number of connections
8
7
6
5
4
3
2
1
0
Without sorting algorithm
With sorting algorithm

Chapter Five: Conclusions

There are many techniques that people use to aid in the retention of memory. Chief among these techniques are the *method of loci* (*memory palace*) and the *link/story* methods. An inspiration for this project was to create the infrastructure that could be used to replicate these techniques in software. Indeed, these techniques were the background for the two overarching goals of the project.

The major focus of the work in this project was in creating software that could create stories from a list of inputted nouns. This was achieved by taking advantage of semantic connections between concepts. To try to make these stories more coherent, it was imperative to create the functionality of sorting a list of nouns in a coherent fashion.

The writer believes that this first goal was admirably achieved. It was shown in Section 4.3 that the software developed is capable of sorting a list of words in a more similar manner to a human than a random method would. Additionally, it was shown in Section 4.4 that the sorting algorithm developed allowed for simpler sentences to be constructed by finding more direct connections between concepts than a random sorting would. It was found to be possible to construct a story from a list of inputted nouns, as was shown by the proof of concept in Section 4.1.

That is not to say that more work could be done on the infrastructure developed. Indeed, an area that could be improved upon is the sorting algorithm designed to sort a list of inputted noun. The approach taken in this work was to iteratively selected the best next word based on how connected it was to the current. However, late into the development process it was considered to use a different approach. This approach would maximise how connected every word pair in the final list is, instead of simply iteratively choosing the best next word. Implementing this new sorting approach was left for further work as it would have involved redesigning a large portion of the software, which would have taken time away from testing the software and preparing results.

The other area of work performed in this project was to create software to perform mappings of noun to a visualisable object. These visualisable objects are all 3D objects. Several approaches were considering. Chief among these approaches was mapping a word to a visualisable object with which it has the most phonetic similarity to. Another approach considered mapping a word to visualisable object that it has a lot of semantic connections to. Examples of such mappings were presented in Section 4.1.

Due to the limited size of the list of visualisable objects used, it was observed that there were many instances of the software not finding any suitable mapping. The writer believes that in further works using the mapping methods developed, the lists of words mappable to should be expanded upon.

The writer believes that the infrastructure developed could have potential applications in testing hypotheses related to the retention of memory. A possible use could be in testing whether software can help improve the retention of information. To do this, a control group must be introduced that are given the list of words to remember with no work done by the software.

Additional groups could receive lists of words as well as processed data from the software. For example, one group might receive the list of words mapped to visualisable objects. Another group could receive a story generated from the list of words and so forth. Testing could then be performed on how many words the users could remember. The hypothesis would be that the people that received extra software generated support remembered more words.

Another application of the infrastructure developed could be in identifying relations between concepts in a processed piece of text. Section 4.2 showed that the software is capable of identifying relationships between nouns that other techniques have not yet found.

The writer is interested to see if the infrastructure developed could have applications in creating software versions of the *method of loci* and the *link method*. Further studies could use parts of what was developed in this work to create such software techniques. It is of great interest to the writer to see what conclusions would come from a study that tested whether software can be used to help in the retention of information.

References

[1] Húhn, Peter el al. (eds): “The living handbook of narratology.” Hamburg, Hamburg University, 2019.

[2] Ryan, James: “Grimes’ Fairy Tales: A 1960s Story Generator, Conference Paper, ResearchGate, 2017.

[3] Klein, Sheldon et al. “Automatic novel writing: A status report. Technical Report #186, Computer Science Department”, The University of Wisconsin, 1973.

[4] Meehan Richard James, “The Metanovel: Writing Stories by Computer”, Research Report, Yale University, 1976.

[5] Pérez y Pérez, Rafael. “MEXICA: A Computer Model of Creativity in Writing. PhD Dissertation”, The University of Sussex, 1999.

[6] Qureshi Ayisha, Manzoor Hana, Rizvi Farwa, “The method of loci as a mnemonic device to facilitate learning in endocrinology leads to improvement in student performance as measured by assessments”, ResearchGate, 2014.

[7] Oaks Tommy, “Storytelling: A Natural Mnemonic: A Study of a Storytelling Method to Positively Influence Student Recall of Instruction”, Doctoral Dissertation, University of Tennessee, 1995.

[8] Robyn Speer, Joshua Chin, and Catherine Havasi. “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”, In proceedings of AAAI 31, 2017.

[9] Princeton University “About WordNet.” WordNet, Princeton University, 2010.

[10] A Gatt, E Reiter, “SimpleNLG: A realisation engine for practical applications.”, Proceedings of ENLG-2009, 2009.