Alasdair Dunbar -ad568@kent.ac.uk Babbling Bard

The babbling bard- A system for generating dialogue in video games

Abstract

A major aspect of world design in video games is the generation of realistic minor characters, with a large aspect of this being how they communicate. Currently most games seem to give NPCs a very limited number of lines due to the time required to design lines for every minor character.

This paper will discuss the Babbling Bard system, a system designed to solve this issue. The system essentially works by selecting phrases from templates and editing them using WordNet, the Jigsaw Bard and Metaphor Magnet depending on the selections made. Alongside this it provides a framework for the conversation to take place in, selecting different types of template depending on how the situation is going. This paper will provide a summary of the system, along with its design and background material as well as the positives, flaws and potential extensions of the program if given more time.

Introduction

The babbling bard system is a Java based conversation generator between two separate NPCs. Its main purpose is to make the generation of minor NPC conversations easier, both by creating a general conversation flow that is likely to be different and mostly realistic each time, and also allowing for hundreds of differently phrased ideas from a limited number of templates. This paper will discuss the inner operation of the system, along with the inspirations behind it.

Background

For the system I was initially inspired by twitter bots such as the joking computer and metaphor magnet, both of which use a template-based system of speech, based upon a corpus of text. I was particularly inspired by metaphor magnets use of irony within the tweets (For example, claiming something is about as 'honest' as a politician), and the joking computers ability to use a very rigid template system which still allowed unique seeming results.

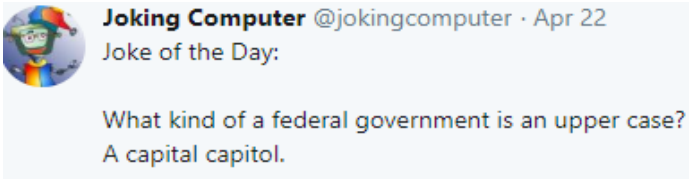

Figure 1: Joking Computer example

I decided upon a template-based text generator as I felt it'd be the most beneficial to game designers, who would need to design the text around their own theme and story, overall, I felt out of most text generation systems it allowed for the most user freedom whilst still generating creative responses. I previously considered Markov text generation but after reading through several examples I felt that the template-based system, whilst being less random overall offered more choice in what was said, which is extremely important from a game design/story standpoint.

Overall, I intended the system to work as a form of exploratory creativity [1] with it being designed to generate sensible creative outputs in relation to the rigid inputs given by the user. Since it is heavily based upon web searches as a corpus one could argue that most of the results aren't historically creative, however due to the transformations of words using WordNet one could argue a lot of the created results are at least personally creative.

WordNet

Wordnet is a large database of lexically linked words developed by the cognitive science laboratory of Princeton University [2]. It is used by many text based creative systems as a means to generate phrases with similar meanings to the input but with far more of a variety on the surface. This at least allows generated text to not seem generic and repetitive. It is also commonly used to find the similarities between words, which could potentially be used more for human inputs to a conversational agent, or to generate partial comparisons.

It is however, limited in certain uses, for example in terms of pronunciation or more figurative uses of the language, and so must be combined with other tools to get more impressive uses out of it other than surface level changes with regards to synonyms.

The Jigsaw Bard

The Jigsaw Bard is a web service developed by the Creative Language System Group of University College, Dublin in order to generate figurative language. This group is also responsible for building Metaphor Magnet, a tool which I will discuss in more detail later. This system is discussed heavily in this Veale's paper on readymade phrases [3] which helped me understand its potential for creativity, explaining the viability of premade phrases within a creative text system, and was partially responsible for inspiring its use within the system.

The Jigsaw Bard relies heavily on Google n-grams (An n gram is a sequence of words from a sample of text comprised of any number of words) in that it uses web searches to build up lists of stereotypes that can then be accessed from the site. This is an extremely powerful system for language processing, as Google searches provides a data set from an extremely large sample of the population.

Overall its best use is the generation of figurative language, with it featuring heavily in my project as a means for generation of similes and ironic insults. Outside of figurative language it is also useful for its large database of co-occurring properties of adjectives, which allows comments to be far more descriptive without just saying the same thing.

Metaphor Magnet and Flux Capacitor

Metaphor Magnet [4] is a web service used to generate conceptually related terms for a related noun/set of nouns. Much like the Jigsaw Bard, Metaphor Magnet heavily relies on google n-grams as a means to generate its responses.

It is used in several well designed natural language generating programs including the twitter bot of the same name, which uses an adapted version of this web service known as Flux Capacitor. Flux capacitor is a particularly interesting example which applies transformations based on differences in terms, so for example changing a scientist into a priest due to their different stereotypes. [5] Alongside this it provides additional context for the transformation in order to better 'sell' the idea. The main example shown for this in the paper is the generated transformation from nun-> prostitute, in which the main character discards religious objects for stereotypical objects on the other end ("Nun hates habits, stockpiles stilettos"). Not only is this a well realized project, but it also provided a good inspiration for my project in terms of using a corpora of data to provide stereotypes for different adjectives/nouns during discussions so they made sense within the context of the template.
The @metaphormagnet twitter bot was also a good inspiration for similar reasons, mostly for showing how one could generate uniqueness within a template form, especially with the sentence "most [nouns] are [adjective] but others are as [adjective] as [oppositeNoun]", which seems to endlessly generate different quotes from such a small template. A similar example is used within the demo templates for my system to showcase possible uses.

Methodology and design

Initial plans

Initially I was planning on creating a fairly basic system which implemented WordNet and mainly used text files as an input, after realising how cumbersome this would get I began to look into web-based solutions. Eventually, I decided to use the Jigsaw Bard, and Metaphor Magnet both of which, as discussed earlier have an extremely large amount of usable language for essentially any commonly used adjective or noun. Text files are still used, but they are mostly used to store user inputs such as templates and planned adjectives and nouns. These templates are editable within these files, allowing the system to be easily extended depending on the theme of the game the user wishes to develop. Templates are selected through a partially random process weighted by character opinions and previous contexts within the conversation.

Alasdair Dunbar -ad568@kent.ac.uk Babbling Bard

Controller class

Conversational order and characters

Currently the system has 3 separate opinion levels that affect how characters will respond to each other, ranging from being hostile, which increases the likelihood of insults and vastly decreases how likely the character is to compliment the other to friendly, which does the opposite. At opinion values in between this the character will have a neutral attitude which encourages a more level response. The system also stores the context of the previous sentence in order to allow the characters to react accordingly to a statement, for example, if previously a template was chosen discussing a threat to their town, the chance for a more neutral discussion increase.

This allows for a natural flow of conversation whilst still allowing the potential for unique ordering, as after all realistically, good conversations have a chance of turning sour, or at least occasionally becoming neutral (And vice versa). Initially character opinions are decided based upon their factions and occupations, with the characters opinions of each other being penalised if they're in separate factions or given a bonus if they're either in the same faction or occupation.

After a form of conversation is chosen a phrase is selected from the list of templates for that particular section. After this, sections of the template are replaced with the appropriate words/phrases. Dependent on the type of phrase the characters opinions will change a random amount (Increasing for positive phrases and decreasing for negative). The conversation will end if the NPCs reach a threshold of hostility to each other, if the conversation repeats or if the conversation has gone on too long.

Generation of text

On a surface level a lot of the changes where created using WordNet, namely in terms of replacing the words chosen with either their synonyms or antonyms depending on the situation, this allowed for a far larger amount of variety within the system, as one manually input adjective can count for many other meaningful words and phrases. With regards to antonyms I also ensured variety by choosing from any of the synonyms of the antonym found (As word net does not reliably return all possible antonyms by itself). For the most part the vast majority of inputs are cycled through word net in order to reduce the risk of typicality. In order to ensure maximum variety I also decided to consider the words generated under the "also sees" function as a synonym as they could all essentially serve the same purpose.

```
Synonym tree of  "generous":
  [PointerTargetNode: [Synset: [Offset: 1115023] [POS: adjective] Words: benevolent, freehearted -- (
  [PointerTargetNode: [Synset: [Offset: 1115129] [POS: adjective] Words: big, bighearted, bounteous,
  [PointerTargetNode: [Synset: [Offset: 1115676] [POS: adjective] Words: lavish, munificent, overgene
  [PointerTargetNode: [Synset: [Offset: 1116182] [POS: adjective] Words: unselfish -- (not greedy)] [
  [PointerTargetNode: [Synset: [Offset: 360539] [POS: adjective] Words: charitable -- (full of love a
  [PointerTargetNode: [Synset: [Offset: 1118176] [POS: adjective] Words: generous -- (not petty in ch
  [PointerTargetNode: [Synset: [Offset: 2106299] [POS: adjective] Words: unselfish -- (disregarding y
  [PointerTargetNode: [Synset: [Offset: 360539] [POS: adjective] Words: charitable -- (full of love a
  [PointerTargetNode: [Synset: [Offset: 1118176] [POS: adjective] Words: generous -- (not petty in ch
  [PointerTargetNode: [Synset: [Offset: 2106299] [POS: adjective] Words: unselfish -- (disregarding y
```

Figure 2:Synonyms of generous
After generating a list of words, an individual is selected at random by shuffling the list and used within the template.

Examples of uses of the Jigsaw Bard

After connecting to the Jigsaw Bards site, the system uses JSoup to collect the tables containing the required lists of data, after editing the lists to remove the bracketed scores these be repurposed into ArrayLists of phrases/words that can then easily be integrated into templates.

As an example, if the template chosen is "you hear about that [threat] at the [location]? [adjective] like [poetic]" it will first of all select the threat and location from a list within the appropriate text files. And then select an adjective from the appropriate list, it will then use a function which sends the adjective to the Jigsaw Bard web service, returning all the poetic comparison phrases. [see figure 3]. The system then uses WordNet to replace the adjective with a related word in order to generate more uniqueness.

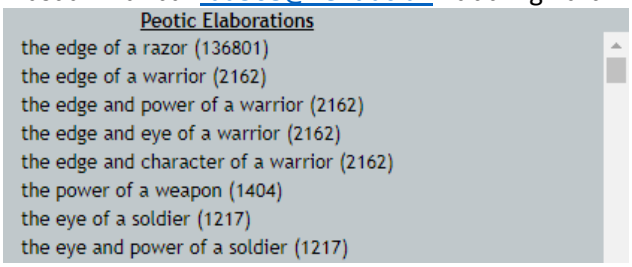

Figure 3 Poetic elaborations of 'Dangerous'

For example, if the selected adjective is “dangerous” the system would send the word to the web service, receive this list and would choose one elaboration at random. It would then replace dangerous with a synonym, generating phrases like “threatening, like the power of a weapon”.

The Jigsaw bard also stores a large number of ironic elaborations, which can be repurposed for insults, for example in the template “you [occupation]s seem to think you’re so [adjective] but in reality, you’re as [secondAdjective] as [ironicNoun]” , the system replaces the first and second adjective with differing synonyms, and then uses the ironic elaborations section to generate something along the lines of “you seem to think you’re so powerful, but in reality, you’re as strong as a timid kitten”. Another use was generating co-occurring properties, that could be used to generate more meaningful phrases, so for example instead of just calling someone intelligent the system is able to return another fitting adjective and call them intelligent and accomplished. I considered using WordNet to replace every word within the poetic elaborations, however I felt that this would reduce the overall quality of the system due to the APIs lack of ability in distinguishing word meanings. I did however use it to generate synonyms for fitting nouns and coAdjectives.

Metaphor Magnet

The process for metaphor magnet was roughly similar except the website is far more suited to returning data, having a specific section for returning XML versions of pages for convenient parsing. Metaphor magnet proved to be a very useful system for figurative language, providing options on whether to be positive or negative dependent on what context one is in (These are not always accurate as shown in the example but for the most part they show promise). Currently the system in place causes any metaphor in neutral or insulting to be negative whilst in the compliment section they count as positive for sake of demonstration.

Figure 4:Positive and negative metaphors for "Leader"

Quality of life improvements- Correct grammar and blacklists.

In order to ensure that all phrases are grammatically correct I created a few functions to check through each phrase generated. With regards to plural terms during the creation of the phrase, if any terms are intended to be plural there is a function that goes through the various grammatical rules for how to create a plural term, creating a mostly accurate final word.

As a result of testing a function was created to fix spacing and a/an placement , the function splits the response into every base word, after this it runs through the response, if the word contains trailing whitespace it trims the word and returns it to the string, if the word was either a/an then the next word in the response is checked and (assuming it exists) if it begins with a vowel “a” would be replaced by an. The opposite occurs for consonants.

In addition, as WordNet, Metaphor Magnet and the Jigsaw Bard contain such a large amount of information they will occasionally generate words that either don't fit the theme of the story (Replacing "dog" with the Latin 'canis familiaris') or are just generally inappropriate and rude.

In order to solve this a blacklist was created, if the generated response contains any word within the blacklist the system generates another phrase. This blacklist is easily editable within a text file, allowing the user to easily customise what they feel should be blocked by just typing in the unwanted words (E.g.: Canis familiaris would be inappropriate in a medieval RPG, but could potentially be useful for a science-based game).

The character class

The character class is used throughout the program as a means for storing information about the two NPCs. It stores their name, faction, occupation and current opinion, alongside methods for setting and returning these variables.

The general UI-User customisation

The program uses a simple JFrame user interface comprised of several JTextBoxes and JButtons for inputs. The program allows the user to create their own characters, with their own factions and occupations in order to test specific responses. It also has specific buttons for generating neutral, insulting or complimentary outputs, which is very useful when testing out specific templates, this is made even more convenient as the provided text files can be changed while the program is running, allowing quick evaluation of quality.

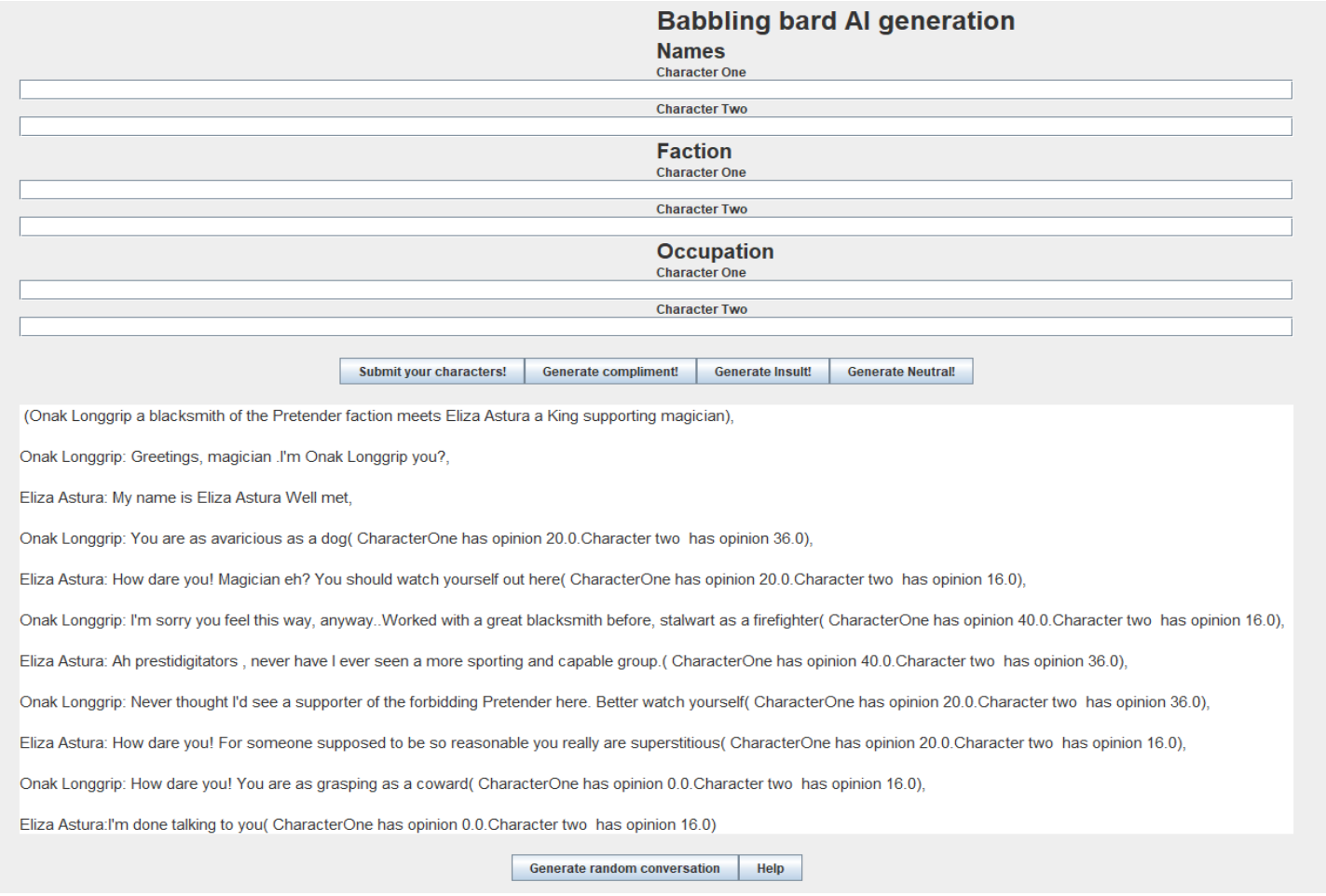

Figure 5 Babbling bard screenshot

The system contains several different text files for user inputs. Ranging from the templates for insults/compliments/neutral responses, to the potential threats, names, occupations, factions and adjectives that the characters can use within the conversation.

The text templates are relatively easy to use, with the user simply typing up what they wish the character to say, with the replaceable sections being surrounded in square brackets for the program to edit at run time. The system comes with both examples and a tutorial that explains their use for user convenience, but for the most part it is intuitive and easy to use.

Evaluation process-Error handling

In order to evaluate my system, I created numerous templates which were then used to check for grammar and general usability. After running through the system a few times, I noticed issues with a/an placement and pluralisation, there were also minor issues with spacing. I fixed the vast majority of these issues as mentioned in grammatical handling but there are a few exceptions to the rule. The a/an solution also does not assist the user when the web services used returns plurals, which is currently a noticeable issue when evaluating creativity for users.

Evaluation process-Creativity

In order to evaluate my system, I mainly used aspects of Graeme Ritchie's criteria for empirical creativity [6]. Namely following his three essential properties of novelty, quality and typicality. I based the questions asked on the informal list of rules from the CO659 lecture on evaluating computational creativity [5]. In particular I evaluated its ability to produce typical and highly valued output, along with it being able to replicate items that inspired the design of the system in the first place (in this case the metaphor magnet twitter bot and general metaphors) whilst also being able to produce item differing from the inspirational system. Alongside this the novel output of this system should also conform to the typicality and usefulness mentioned previously.

In order to ensure I wasn't being biased when evaluating my system, I asked two other people to evaluate the system, the following sections will contain my responses, and summarise the responses of the other users.

A decent proportion of the output should be suitably typical

I personally feel that for the most part the outputs were suitable for the system and tended to follow at least a general pattern that seemed to work within a conversation. Overall it was agreed that the typicality of the quotes was acceptable, the main issues within the system were due to issues within quality.

A decent proportion of the output should be highly valued for the task

In my opinion, there were a large amount of outputs that were suitable for the task and worked as genuine cases of figurative language. The proportion of high quality texts depended on the templates chosen, with some being far better on average than others, however I feel for the most part that even in the worst cases at least half of the outputs were satisfactory.

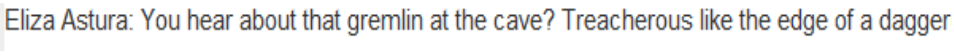

Figure 6:Example of higher valued text generated by the system, uses a mixture of WordNet and the Jigsaw Bard

There were however some issues, especially with the WordNet section for generating antonyms. This was mostly due to the limitations of the JWNL API, which is unreliable in terms of what adjectives had antonyms, and in several cases when using built in methods to return a wider range of antonyms, would return *synonyms* of the adjective. In cases like this in which there are no matching words/phrases that the API/web service can find, the system will return the same word, generating invalid sentences ("For someone supposed to be funny, you really are funny"). These errors however only occur with specific user inputs and can be avoided through user testing of the system.

During the evaluation stage of the system both users testing agreed that many of the sentences were well made, however they felt that there were several outputs that were inappropriate, or uninspiring for the task. As a result of this I added the blacklist, which is explained in more detail above.

There were also cases where there was too much variance in terms, for example when replacing various occupations with their synonyms the program would output something of a completely different meaning (I.e.: Wizard being replaced with ace due to the association with skill). There were also a few issues with metaphor magnet, which

Alasdair Dunbar -ad568@kent.ac.uk Babbling Bard
occasionally generates positive adjective pairs within the section designed around negativity and vice versa, whilst this is mainly affected by the web services ability to tell between positive and negative it still affects user perceptions of the systems creativity.

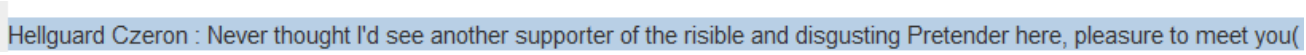

Figure 7: The jigsaw bard contains a large portion of co-adjectives, but does not distinguish between positive and negative meanings.

The system can replicate many artefacts that guided construction of the system

This system can certainly replicate figurative language in the form of a conversation. The main inspiration for this system was metaphor magnet, who's figurative speech templates can be replicated to a degree, namely ones where it uses ironic language.

The users I asked to test the system were not familiar with metaphor magnet, but as mentioned earlier felt that they could see many of the phrases generated being used within an RPG.

Much of the output of the system is not in the inspiring set, and so is novel to the system

The babbling bard is not wholly reliant on the same corpus of text as metaphor magnet, and, and therefore will produce outputs unrelated to the bot. Alongside this the templates generated are for the most part separate from the ideas of metaphor magnet, and so potentially any level of differing ideas could be created.

Novel output of the system should be suitably typical and highly valued

I believe that outside of the templates making use of the metaphor magnet web service the vast majority of the results would count as novel, and I would refer to the above sections referring to this. I personally believe any of the

Limitations to the systems creativity

As mentioned in the background there are major limitations to the program based around its general format as a template-based text generator, leading to it to be unable to produce anything truly unique (Historically creative) outside of its purpose. However, this is in part intentional, game developers may not want dialogue to be completely off-track. Alongside this, whilst it was not the purpose of the program the system lacks the ability to 'appreciate' the value in its art and cannot evaluate whether its results are any good. This was not intended to be a feature of the system, but it is worth noting, as many models of creativity (For example, the tripod model of creativity) take this into account as a vital feature of computational creativity.

Extensions

If given more time I believe there are many extensions that could be made to this project, being a conversational generator at heart the more possible forms of conversation the better. First and foremost, the ability to make simple jokes in a similar manner to the joking computer would be important, as it would fully make sense in the realm of idle NPC conversation.

Alongside this it may be useful to have a way to choose which synonyms are more useful, in terms of the user being able to rate individual phrases, with the likelihood of a word being chosen being directly proportional to its score. This could prevent relevant yet uninspiring words/phrases from being selected as often. I also feel that having an option for users to set the minimum/maximum level of opinion change per comment would be useful, alongside having control over the initial overall opinion.
Finally I feel that creating a more in depth grammatical test would be useful, potentially checking for exceptions for plural terms and the an/a rules.

Conclusion

In conclusion, I feel that the program served its task at generating unique template phrases to an acceptable level, I certainly feel that there were elements that could be extended, especially in making certain the grammar is correct, however overall the system is usable enough that it serves its purpose as a generator with human supervision.

Alasdair Dunbar -ad568@kent.ac.uk Babbling Bard

References

All PDF's accessed 07/05/2018

[1] Boden, M (2007) Creativity, how does it work? [pdf]: https://pdfs.semanticscholar.org/120e/b04b9b69b5f892904a2f6870b8c04cb33f82.pdf

[2] Fellbaum, C (2008). WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press

[3]Veale,T & Hao ,Y (2011)Exploiting Readymades in Linguistic Creativity: A System Demonstration of the Jigsaw Bard[pdf] https://pdfs.semanticscholar.org/43e6/c1a0735eb1a9346826461b2f08ca586b8d68.pdf

[4] Veale T, & Li, G (2012) Specifying Viewpoint and Information Need with Affective Metaphors[pdf]: http://afflatus.ucd.ie/Papers/ACL%20Veale%20and%20Li%202012.pdf

[5] Veale, T (2014) Coming Good and Breaking Bad: Generating Transformative Character Arcs For Use in Compelling Stories[pdf] http://afflatus.ucd.ie/Papers/ICCC2014.pdf [Accessed 07/05/2018]

[6] Ritchie, G (2007) Some Empirical Criteria for Attributing Creativity to a Computer Program[pdf] http://axon.cs.byu.edu/~dan/673/papers/ritchie.pdf

[7]Jordanous, A (2018) Evaluating computational creativity-C0659, University of Kent